CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

AIRCRAFT ARTIFICIAL HORIZON
WITH
INTERFACE TO 3D ANDROID APPLICATION

A graduate project submitted in partial fulfilment of the requirements
for the Degree of Master of Science
in Electrical Engineering

By

Robert Zdunski

May 2012

The graduate project of Robert Zdunski is approved:

Ramin Roosta, Ph. D.	Date
Xiaojun Geng, Ph. D.	Date
Ronald W. Mehler, Ph. D., Chair	Date

California State University, Northridge

Dedication

To Ting,

My dearest wife, thank you for your love and support.

Acknowledgements

- I would like to give special thanks to Professor Ronald Mehler for agreeing on supervising this project, his dedication and contribution in providing quality education in Electrical Engineering program as well as for his patience and help in shaping this project throughout its course.

- I would like to thank Professor Ramin Roosta and Professor Xiaojun Geng for agreeing on serving as a Graduate Project Committee Members and their support.

- Thanks to the Department of Electrical and Computer Engineering for an excellence in providing a quality program and creating great environment that encourages pursuing an engineering degree.

Table of Contents

Signature Page ii
Dedication iii
Acknowledgements iv
Table of Contents v
List of Figures vii
Abstract xi
Chapter 1 1
  Introduction 1
    Overview. 1
Chapter 2 3
  Design Overview 3
    Design elements 3
    Android device 3
    Gyroscope/accelerometer board 6
Chapter 3 10
  Aircraft Attitude Indicator Implementation 10
    Introduction to Attitude Indicator and its features 10
    Motion and coordinate systems fundamentals 12
    Android platform 14
    Gyroscopic chip description 18
    Embedded implementation 20
    Gyroscope/Accelerometer readings processing 23
Chapter 4 29
  Android Application 29
    Overview of Android platform and related application development process and tools 29
    3D graphics development 31
    Features of Inflexion UI 32
    Inflexion UI installation 33
Chapter 5 52
  Development of 3D Objects and their integration with Inflexion UI 52
    3D Format 52
    3D IDE 53

Using 3D objects within Inflexion UI environment 59
Chapter 6 65
USB Interface 65
Overview 65
ADB (Android Debug Bridge) 70
Debugging feature of the Android device 72
Main components of the Android implementation of the USB device ADB server 72
Embedded implementation of the USB host / ADB client 73
Chapter 7 75
Conclusions 75
References 77
Appendix A 79
Development Software Installation Guide 79
Appendix B 95
Schematic & PCB 95
Appendix C 97
Bill of Materials 97
Appendix D 99
Android Application Source Code 99
Appendix E 100
Inflexion Source 100
Appendix F 101
Firmware Source Code 101

List of Figures

Fig. 1) The screen depicting some of aircraft's control instruments (pitch=0°; bank=15°) [1] 2
Fig. 2) i.MX53 [1] 4
Fig. 3) Tap "ZERO" to zero-out attitude indicator (force pitch and bank to equal 0) 5
Fig. 4) Tap marked area to turn on/off precise filter coefficient setup 5
Fig. 5) Tap marked area to turn on/off rough filter coefficient setup 5
Fig. 6) PIC24FJ256GB106 pinout [2] 7
Fig. 7) MPU-6000 pinout [5] 8
Fig. 8) Orientation of MPU-6000 axes of sensitivity and polarity of rotation [3] 8
Fig. 9) Typical Vacuum Attitude Indicator [15] 10
Fig. 11) Interpretation of Attitude Indicator readings [14] 12
Fig. 12) Rotations of the object defined by Euler angles [12] 13
Fig. 13) Earth-Centrered reference frame [13] 14
Fig. 14) Earth-Fixed reference frame [13] 14
Fig. 15) Local-Horizontal reference frame [13] 14
Fig. 16) Attitude Indicator App running on i.MX53 development board with IMX28LCD touch screen 15
Fig. 17) Embedded MotionApps platform by Invensense [19] 17
Fig. 18) Serial interfaces location in the scope of the design modules 18
Fig. 19) MPU-60X0 axes orientation 18
Fig. 20) Embedded circuit schematic 20
Fig. 21) Embedded circuit PCB (bottom side) 21
Fig. 22) Embedded circuit PCB (top side) 21
Fig. 23) X-axis gyroscope driven mode [22] 23
Fig. 24) Complementary filter – the concept 25
Fig. 25) Complementary filter implementation 26
Fig. 26) Invensense Motion Processing platform 27
Fig. 27) Axes orientation with reference to the Android device 27
Fig. 28) Inflexion UI solution in the scope of Android platform [28] 32
Fig. 29) Inflexion UI installation: Project import 33
Fig. 30) Inflexion UI: Selection of the active project 34
Fig. 31) Inflexion UI: Location of the Collada files 35
Fig. 32) Inflexion UI: The placement of the 3D objects on the layout 35

Fig. 33) Inflexion UI: The previewer 36
Fig. 34) Inflexion UI: Access to element’s properties 37
Fig. 35) Inflexion UI: Setting a draw plane 37
Fig. 36) Inflexion UI: Basic properties of the 3D object 38
Fig. 37) Inflexion UI: The layout 38
Fig. 38) Inflexion UI: Layout’s condition 39
Fig. 39) Inflexion UI: Local variables 40
Fig. 40) Inflexion UI: Shared variables 41
Fig. 41) Inflexion UI: Module registration 42
Fig. 42) Inflexion UI: Java class 44
Fig. 43) Inflexion UI: Inflexion framework imports 45
Fig. 44) Inflexion UI: Class member 45
Fig. 45) Inflexion UI: Integration of the module implementation class with Android app 46
Fig. 46) Inflexion UI: Project build 47
Fig. 47) Inflexion UI / Eclipse integration – “jni” folder in Eclipse 48
Fig. 48) Inflexion UI / Eclipse integration – “jni” folder in Inflexion 48
Fig. 49) Inflexion UI / Eclipse integration – “sample packages” folder in Eclipse 49
Fig. 50) Inflexion UI: Package wizard 49
Fig. 51) Modelling in Blender v. 2.5 54
Fig. 52) Editable elements of the 3D objects [33] 54
Fig. 53) UV mapping of the cube 3D object [32] 55
Fig. 54) 3D object texturing through UV mapping [32] 55
Fig. 55) Attitude Indicator 3D representation created in Blender v. 2.5 with 2D picture used to UV map the sphere part of the 3D object 56
Fig. 56) Partial (sphere) mapping of the Attitude Indicator 3D object (imported by Blender v. 2.49) 57
Fig. 57) Partial (ring) mapping of the Attitude Indicator 3D object (imported by Blender v. 2.49) 57
Fig. 58) Partial (wings) mapping of the Attitude Indicator 3D object (Blender v. 2.49 import is shown in Fig. 59) 58
Fig. 59) Export of the 3D object from Blender v. 2.49 to Collada (.dae) file using settings acceptable by Inflexion UI (shown) 59
Fig. 60) 3D objects available from within Inflexion UI (.dae and .png files under “Project Explorer”) 60
Fig. 61) Primary 3D object properties available in Inflexion UI 61
Fig. 62) Placement of a “Touch Region” component on the active layout using “Element

Manager” 62
Fig. 63) Access to “Touch Region” properties through “Element Manager” 62
Fig. 64) Displacement setup access through “Layout Manager” 63
Fig. 65) The assignment of the variable “pitch” to drive a displacement in the range of -1800 ÷ 1800, i.e. 360deg in both directions with 0.1 degree resolution for smooth transitions within a real range of -180 ÷ 180 degrees. 63
Fig. 66) Linking “pitch” displacement with “gyro_sphere” 3D object and defining 360 degrees rotation of the object around its X axis to be controlled by the variable previously assigned to that displecement. 64
Fig.67) Inflexion Previewer 64
Fig. 68) Android device as a USB device (a). Android device as a USB host (b). [35] 66
Fig. 69) USB driver architecture [35] 67
Fig. 70) USB embedded host state machine - Format of a single USB transfer [34] 70
Fig. 71) Chain of interfaces utilized in the Aircraft Attitude Indicator application 71
Fig. 72) Linux shell and file system access on Android device through Windows “cmd” and ADB service 71
Fig. 73) ADB launch through Windows’ “cmd” 71
Fig. 74) Access to Android device serial number via ADB protocol through Windows “cmd” and through Eclipse 72
Fig. 75) Java version check 79
Fig. 76) ADT plugin install details 81
Fig. 77) ADT preferences 82
Fig. 78) [41] Last historical dataset collected during a 14-day period ending on February 1, 2012 (based on the number of Android devices that have accessed Android Market) 83
Fig. 79) Android SDK Manager 83
Fig. 80) Cygwin: awk setup 84
Fig. 81) Cygwin: make setup 85
Fig. 82) Inflexion UI: Express/Runtime installer launch 85
Fig. 83) Inflexion UI Runtime: Target error message 87
Fig. 84) Project properties automatic fix 88
Fig. 85) Run configuration 88
Fig. 86) i.MX53-QSB-Android-Gingerbread-Release4.0 extraction 90
Fig. 87) SD (Secure Digital) as a Linux device 91
Fig. 88) "flash_prebuilt_android.sh" script permission setup 91
Fig. 89) "flash_prebuilt_android.sh" script execution 92
Fig. 90) "flash_prebuilt_android.sh" script execution progress 92

Fig. 91) MPLAB Toolsuite setup 94
Fig. 92) MPU-6000 board circuit schematic 95
Fig. 93) MPU-6000 PCB: Bottom side 96
Fig. 94) MPU-6000 PCB: Top side 96
Fig. 95) BOM 98

Abstract

AIRCRAFT ATTITUDE INDICATOR WITH INTERFACE TO 3D ANDROID APPLICATION

By

Robert Zdunski

Master of Science in Electrical Engineering

The goal of this project is to provide a 6-axis Aircraft Attitude Indicator solution based on Android OS and cutting edge MPU-6000 chip by Invensense. Mentor Graphics emerging Inflexion UI technology to create 3D environment for Android devices has been utilized on i.MX53 platform. Motion-fusion algorithms have been developed as well as Invensense's proprietary and encrypted MotionFusion™ library has been tested. The project involves tools like Inflexion, Eclipse, Blender, MPLAB and C/C++ JAVA languages to create complete software solution. Hardware includes i.MX53 and external board designed for this project with microcontroller from Microchip with USB capabilities and MPU-6000 populated on it. USB and I^2C interfaces have been implemented.

Chapter 1

Introduction

1.1. Overview.

This report addresses implementation of the aircraft’s artificial horizon on the Android Operating System. "The first attitude instrument (AI) was originally referred to as an artificial horizon, later as a gyro horizon, now it is more properly called an attitude indicator." [42] Therefore the name "Artificial Horizon" will be used interchangeably with an "Attitude Indictor" throughout this report. The term "Artificial Horizon" refers to the line that separates the sky (blue) and the ground (brown) which is supposed to reflect the actual position of the aircraft in relation to the real horizon seen from the perspective of the pilot, while the term "Attitude Indicator" refers to the same spatial position of the aircraft defined by pitch and bank, but also may include Earth's magnetic-north related heading indicator. Aircraft’s attitude indicator is one of the flight instruments installed in the cockpit of the aircraft, whose purpose is to provide the pilot with information about the current spatial position of the machine without visually referencing to the outside objects. Flight instruments are divided into categories: control instruments (e.g. power), performance instruments (e.g. altitude, speed), navigation instruments (e.g. GPS-Global Positioning System). Basic flight instruments are airspeed indicator, attitude indicator, altimeter, turn coordinator, heading indicator, and vertical speed indicator [2]. Attitude indicator falls under "Control Instruments" category, shown in Fig. 1. The basic requirement that applies to attitude indicator is that it must be properly installed on the machine, i.e. it must be attached so that its pitch-sensing axis is perpendicular to aircraft’s fuselage as it indicates the position of the object that needs to share exactly the same system of coordinates.

"Until recently, most general aviation aircraft were equipped with individual instruments utilized collectively to safely manoeuver the aircraft by instrument reference alone. With the release of the electronic flight display system, the conventional instruments have been replaced by multiple liquid crystal display (LCD) screens." [1] In this project widely available Android device has been used instead of multiple LCD screens with interface to Attitude Indicator. Additional instrumentation may be added to the solution and communicate with 3D UI via USB interface in the future. After Android devices with USB host interface become widely available on the market, the external Attitude Indicator board could be powered from that device’s battery in case of aircraft’s electrical failure. Currently available Android development tools enable the design of outstanding 3D graphics that would fit individual requirements and would constitute easily modifiable and standardized platform.

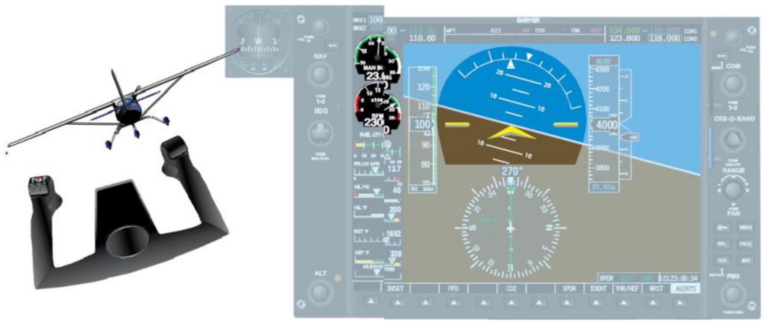


Fig. 1) The screen depicting some of aircraft's control instruments (pitch=0°; bank=15°) [1]

In order to obtain actual position of the Aircraft in relation to Earth's gravity, i.e. its pitch and bank, either gyroscopic device or accelerometer or both are required to be incorporated into the system. Albeit, when aircraft's pilot is maintaining constant altitude he uses altimeter as the primary instrument to control the pitch and as long as the machine maintains constant airspeed and pitch attitude, the altitude remains constant. Also, the pilot relies on the heading indicator to lean about current bank when flying in instrument meteorological conditions. In case, both, accelerometer and gyro sensor are used, their outputs need to be combined in so called Motion-Fusion process, which calculates estimated output based on readings from gyroscope and from accelerometer rather than trusting one device only. This approach improves the quality of final result by eliminating drawbacks of the sensors and by emphasizing their strengths at the same time. To do so algorithm utilizing complementary filter solution has been employed.

The MPU-6000 by Invensense has been picked as a 6-axis gyroscope/accelerometer MEMS (MicroElectroMechanical System) chip due to its extraordinary feature of delivering 6-axis MotionFusion™ data encompassing 3-axis gyroscopes and 3-axis accelerometers fabricated on the same die, thus eliminated PCB level misalignments, with MPU (Motion Processing Unit) and optimized for 8-bit embedded microcontrollers with limited MIPS and memory. MPU-6000 comes with MotionApps software platform designed to help in shortening time to market.

Chapter 2

Design Overview

2.1. Design elements.

The design consists of the following elements:

- Android device: i.MX53 development board provided by freescale with Android BSP (Board Support Package) from Adeneo.

- 3D user interface created using Blender and Inflexion UI from Mentor Graphics and Android application developed through integrating Inflexion UI outputs with Android application JAVA code in Eclipse IDE.

- Custom, external board designed for this project with MPU-6000 chip from Invensense and PIC24FJ256GB106 microcontroller from Microchip. I^{2}C interface on board is used as a communication channel between MPU-6000 and PIC24FJ256GB106. USB interface on board is used as a communication channel between PIC24FJ256GB106 and i.MX53.

- PIC24FJ256GB106 firmware compiled using C30 compiler and programmed using MPLAB tool, both provided by the chip’s manufacturer, i.e. Microchip.

2.2. Android device

- i.MX53 board, shown in Fig. 2, with IMX28LCD (WVGA) touch-screen (optionally VGA output can be used to drive VGA monitor => no touch screen):

  - CPU: ARMCortex-A8 1GHz+ multi-core
  - Process:65nm, LP/GP
  - Core Voltage:0.85V-1.3V
  - Package:19x19 0.8mm 529 ball BGA12x12 0.4mm PoP (Consumer)
  - Case Temp:-20 to 70C (Consumer)-40 to 85C (Auto/Industrial)
  - 2500 –14,000+ DMIPS
  - Graphics: Adv 2D+3D HW
  - Full HD capability
  - Display up to UXGA (1600x1200)
  - Video: >1080p enc/dec
  - LCD: >1080p
  - PMIC: Integrated/Separate
  - HDD: PATA, S-ATA interface
  - One eSDHC ports supports MMC4.4 including DDR mode
  - Delivers rich graphics and UI in HW
  - OpenGL™ ES 2.0 3D accelerator (AMD Z430)
  - OpenVG™ 1.1 graphics accelerator (AMD Z160)

- NEON™ Vector floating point co-processor
- Open source development platform
- Connectivity:

  - High speed USB OTG and HS Host, with embedded Phy(s) (2x). HS Host x2
  - Up to 800Mbps LP-DDR2, LV-DDR2, DDR2 & DDR3, 2GB total DDR.
  - SLC/MLC NAND Flash 8/16-bit, up to 16-bit ECC
  - SRAM/NOR
  - Ethernet 10/100 with IEEE1588 HW enabled
  - High speed eMMC 4.3/4.4, SD 2.1, UART, SPI
  - ATA-6, SATA 2 + PHY
  - 3.3V and GPIO support on most non-DDR pins

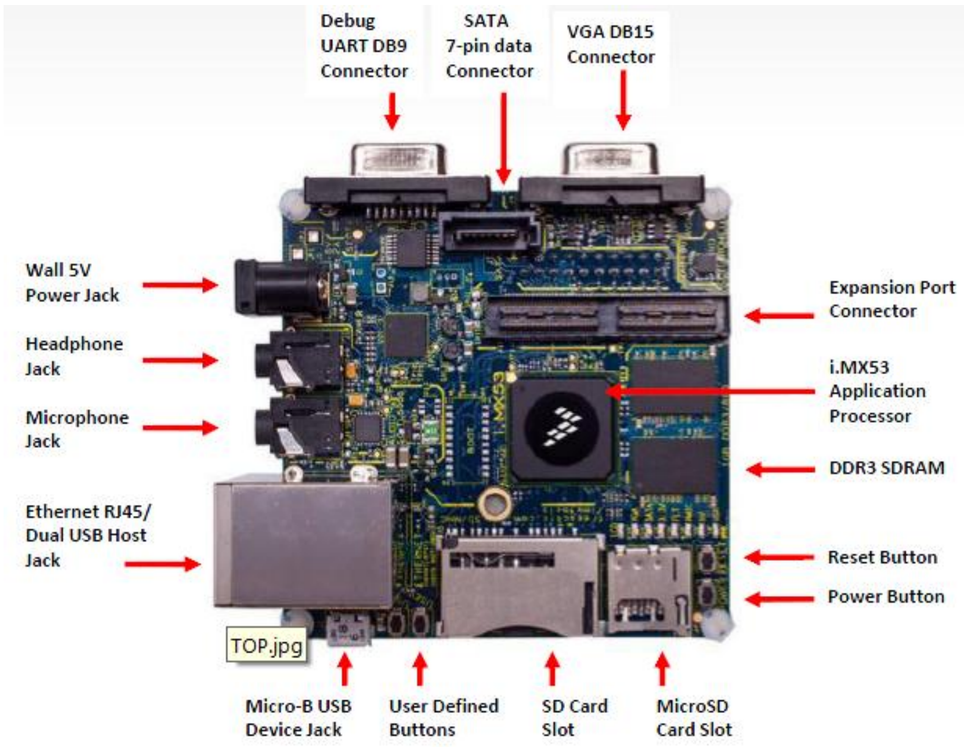


Fig. 2) i.MX53 [1]

- Android application operation manual illustrated in Fig. 3-5.

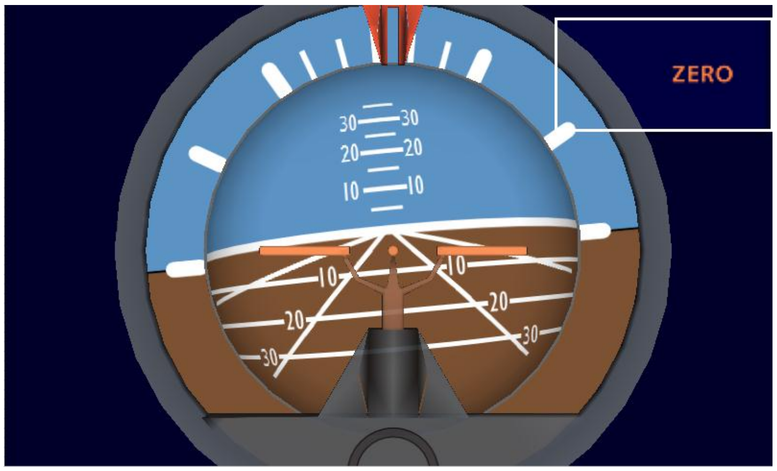


Fig. 3) Tap “ZERO” to zero-out attitude indicator (force pitch and bank to equal 0)

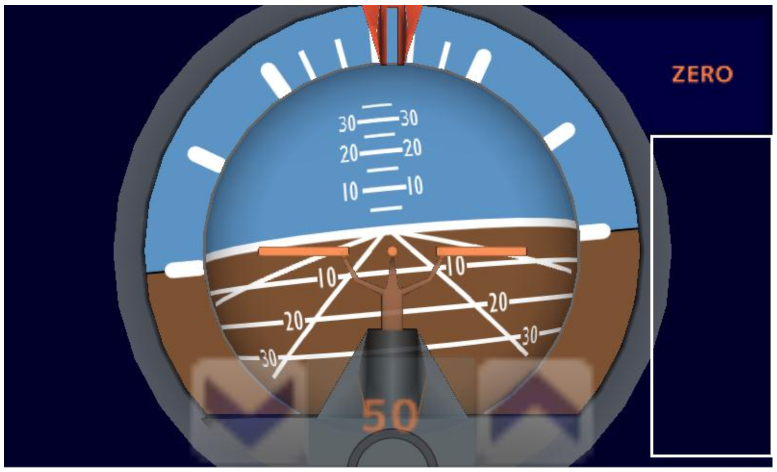


Fig. 4) Tap marked area to turn on/off precise filter coefficient setup

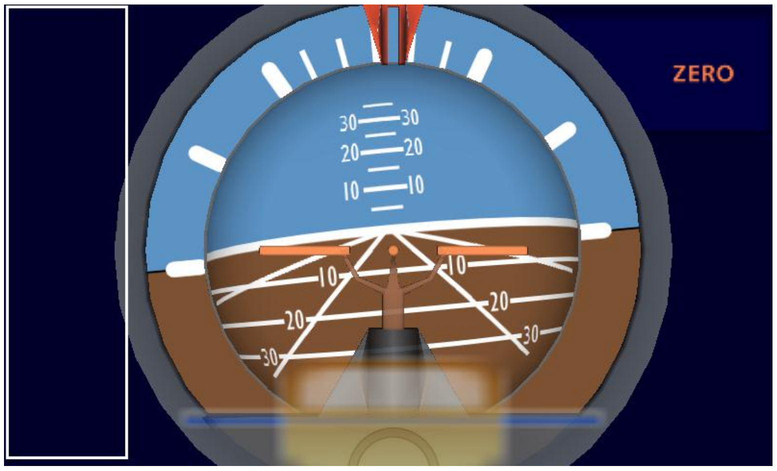


Fig. 5) Tap marked area to turn on/off rough filter coefficient setup

Settings are saved on the SD card inserted into SD mini socket on i.MX53 board and valid after power down/up procedure.

2.3. Gyroscope/accelerometer board.

- PIC24FJ256GB106 microcontroller shown in Fig. 6 with I^2C interface to MPU-6000 and USB interface to i.MX53.

    - Modified Harvard Architecture
    - Up to 16 MIPS Operation at 32 MHz
    - •8 MHz Internal Oscillator
    - •17-Bit x 17-Bit Single-Cycle Hardware Multiplier
    - •32-Bit by 16-Bit Hardware Divider
    - •16 x 16-Bit Working Register Array
    - •C Compiler Optimized Instruction Set Architecture with
    - Flexible Addressing modes
    - •Linear Program Memory Addressing, Up to 12 Mbytes
    - •Linear Data Memory Addressing, Up to 64 Kbytes
    - •Two Address Generation Units for Separate Read and
    - Write Addressing of Data Memory
    - Connectivity:

        - USB:

            - USB v2.0 On-The-Go (OTG) Compliant
            - •Dual Role Capable – can act as either Host or Peripheral
            - •Low-Speed (1.5 Mb/s) and Full-Speed (12 Mb/s) USB
            - Operation in Host mode
            - •Full-Speed USB Operation in Device mode
            - •High-Precision PLL for USB
            - •Internal Voltage Boost Assist for USB Bus Voltage
            - Generation
            - •Interface for Off-Chip Charge Pump for USB Bus
            - Voltage Generation
            - •Supports up to 32 Endpoints (16 bidirectional):
            - USB Module can use any RAM location on the device as USB endpoint buffers
            - On-Chip USB Transceiver with On-Chip Voltage Regulator
            - •Interface for Off-Chip USB Transceiver
            - •Supports Control, Interrupt, Isochronous and Bulk Transfers
            - •On-Chip Pull-up and Pull-Down Resistors

        - UART w/ IrDA
        - SPI
        - I^2C

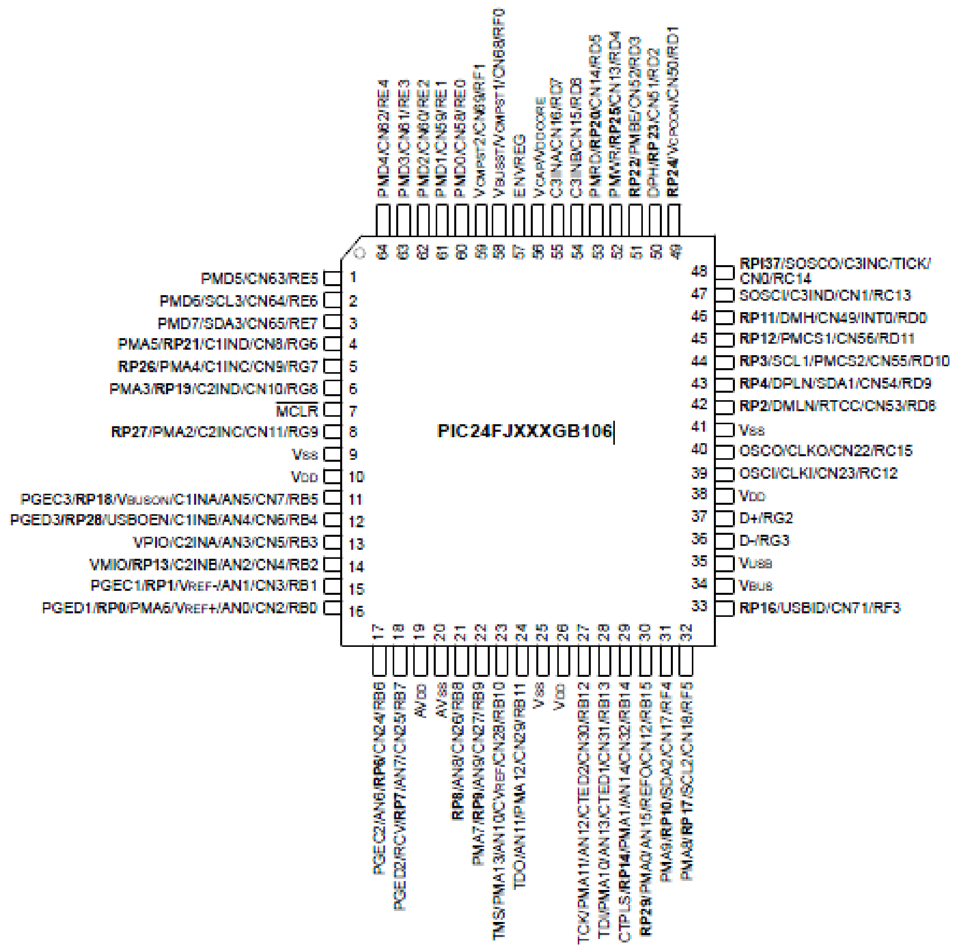


Fig. 6) PIC24FJ256GB106 pinout [2]

- MPU-6000 gyro/accel chip by Invensense shown in Fig. 7. Fig. 8 illustrates axes orientation.
  - Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec
  - Digital-output tri-axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g
  - Data is measured using on-chip ADCs and transmitted over I²C or SPI interface
  - VDD 2.5V±5%, 3.0V±5%, or 3.3V±5%

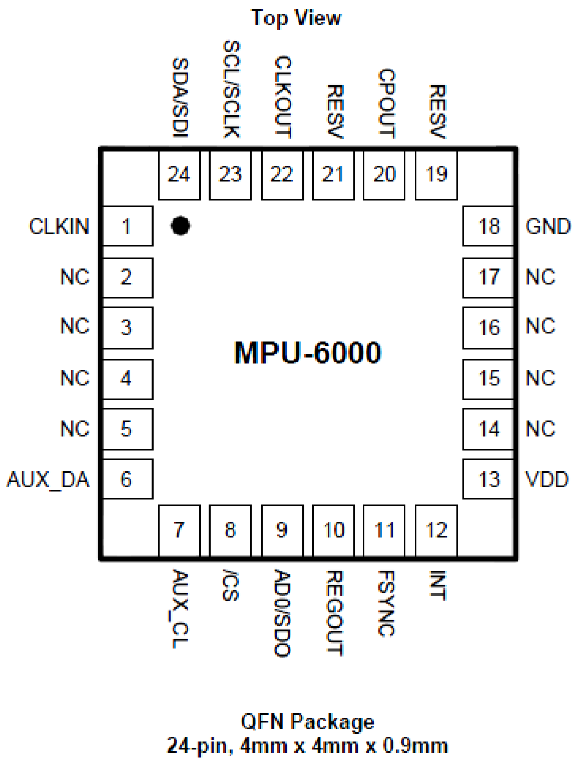


Fig. 7) MPU-6000 pinout [5]

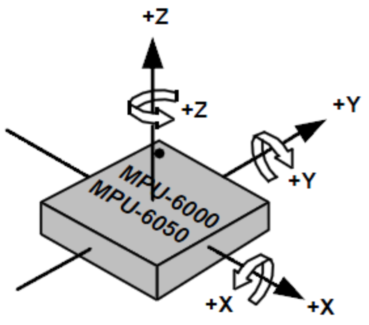


Fig. 8) Orientation of MPU-6000 axes of sensitivity and polarity of rotation [3]

- RJ-11 ICSP (In Circuit Serial Programming) microcontroller programming connector (ICD-3 programmer by Microchip used to update the firmware).

- USB-MINI-B 5VDC power connector.

- USB-A communication interface port.

- 100-mil pitch UART connector (needs UART - RS-232 or UART – USB converted to interface with PC).

- SMD power LED indicator.
- Through-hole No-Motion LED indicator.

Chapter 3

Aircraft Attitude Indicator Implementation

3.1. Introduction to Attitude Indicator and its features.

Aircraft’s Attitude Indicator provides the pilot with the rotation around the longitudinal axis to indicate the degree of roll (bank), and around the lateral axis to indicate the pitch (nose up/down).

In a typical vacuum, shown in Fig. 9, installed in the circuit similar to the one depicted in Fig. 10, or electrical implementation of the Attitude Indicator the rigidity of rotating gyroscope is used. Due to its inherited characteristics, the gyroscope maintains fixed position regardless the aircraft's attitude.

The pilot focuses on the following parts of the Attitude Indicator:

- The miniature wings mounted on the indicator's casing. Those wings indicate the position of the real wings of the aircraft in relation to its spatial position (miniature wings are mounted in parallel to real wings).
- The horizon line that separates the top (sky) and the bottom (ground) parts of the indicator
- The position marks that indicate actual number of degrees the aircraft's attitude changes about since last zero out procedure performance.

The reference arm, with its attached blue and black card representing the sky and ground respectively, remains upright relative to the actual horizon as the airplane climbs, descends, and banks. [15] "Vacuum pressure draws fast moving air alongside the cupped edges of this gyroscope causing it to spin. This spinning gyroscope remains rigid in space, regardless of the pitch or bank attitude of the airplane."

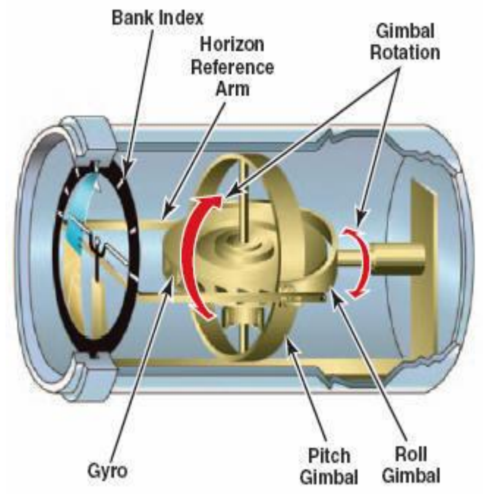


Fig. 9) Typical Vacuum Attitude Indicator [15]

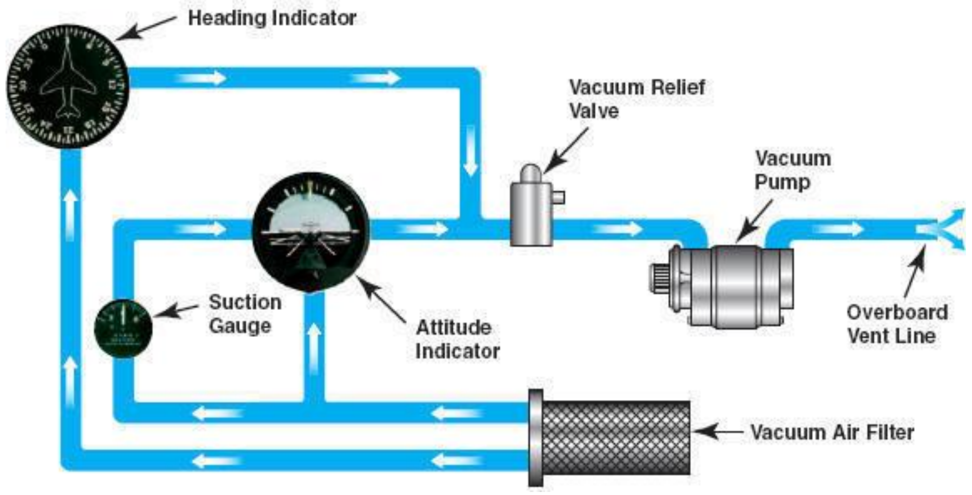


Fig. 10) A typical pneumatic circuit of the Attitude Indicator [15]

The alignments of the miniature wings with the horizon bar indicates that aircraft is in the flight leveled in relation to its position at the time the last zero out operation was performed. Whenever miniature wings are located above the horizon line, it indicates that the aircraft is climbing, and vice versa, the wings below the horizon line indicate descending of the aircraft.

The sky is usually blue and the ground is usually brown in most commercially available solutions.

The rigidity of the gyroscope maintains the horizon line in parallel to the natural horizon, provided the last zero out procedure was performed with the gyroscope in level with the natural horizon. Zero out procedure always sets the reference that might not necessarily be a natural horizon. The miniature wings representing a real aircraft move with the object the attitude indicator is attached to, i.e. aircraft itself. The movement ratio indicates the degree of pitch and bank changes.

For this project electronic the Aircraft's Attitude Indicator that communicates with Android application has been designed. This device uses gyroscope as well as accelerometer to calculate actual spatial position of the aircraft in relation to its position at the time of zeroing out. Fig. 11 illustrates interpretation of the Attitude Indicator readings performed by the pilot.

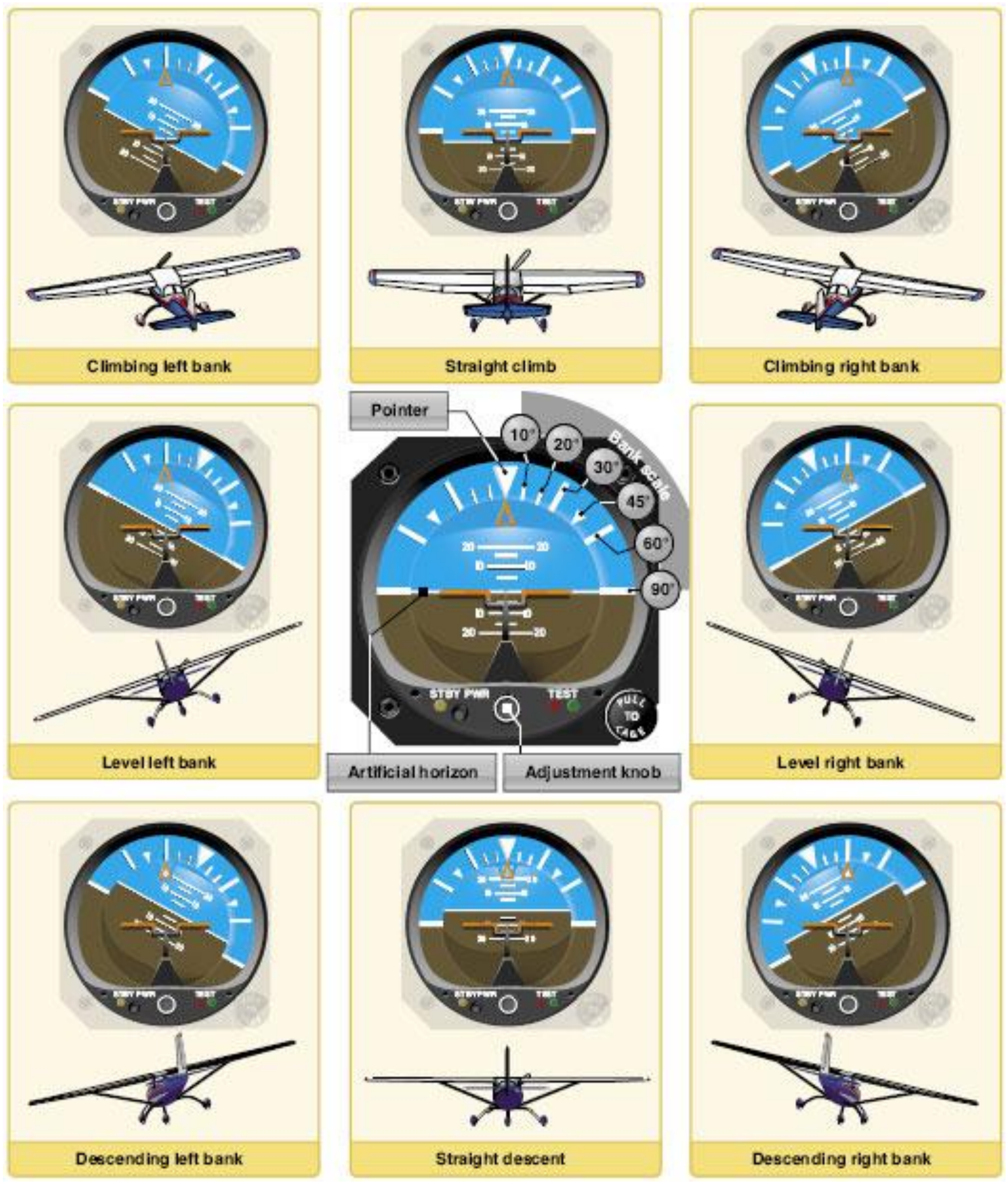


Fig. 11) Interpretation of Attitude Indicator readings [14]

3.2. Motion and coordinate systems fundamentals.

Coordinate systems are used to keep track of the relative object's position and orientation in space. The simple way to model an aircraft's coordinate system is to use the one which is fixed in the body of the aircraft itself. Then, the forward direction is modified by the presence of wind, and the motion of the object through the air is not the same as its motion relative to the ground. [12]

- ❖ Body-fixed frame of reference

  - The orientation of the body coordinate axes is fixed in the shape of body and the aircraft is assumed to be rigid.

  - First of the axes points through the nose of the craft, the second is perpendicular to the first and the third is perpendicular to the first/second axes plane and points down through the aircrafts bottom.

  - The origin of the Frame (Body) coordinate system perpetually moves with the object it’s attached to.

  - In the body axis coordinates system, where the axes are fixed to the rigid object's body, aerodynamic forces and moments depend upon relative velocity orientation angles, hence are not referenced to the earth axes.[13]

  - Rotational degrees of freedom are defined by quaternions, rotation matrix or Euler angles [12] shown in Fig. 12:

▪ P or Φ	Roll about the x axis
▪ Q or Θ	Pitch about the y axis
▪ R or Ψ	Yaw about the z axis

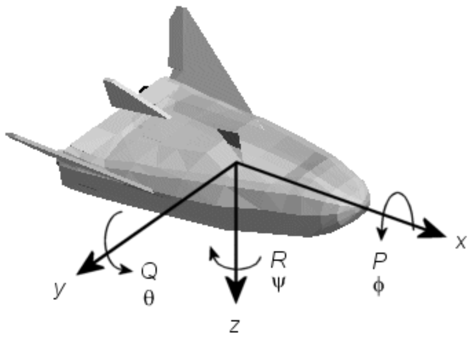


Fig. 12) Rotations of the object defined by Euler angles [12]

- World-fixed coordinates systems illustrated in Fig. 13-15.
  The origin of the system is fixed to an arbitrary point on the surface of the Earth.

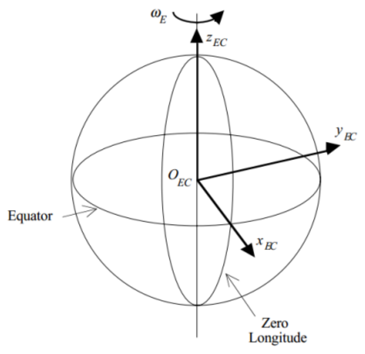


Fig. 13) Earth-Centrered reference frame [13]

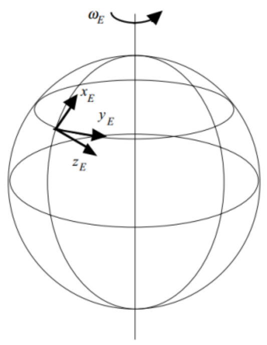


Fig. 14) Earth-Fixed reference frame [13]

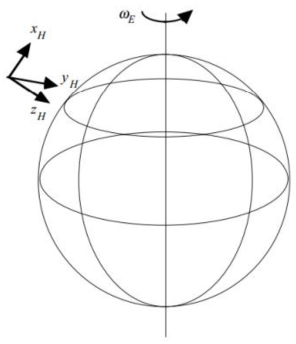


Fig. 15) Local-Horizontal reference frame [13]

3.3. Android platform

- Android application visualizes aircraft dynamics by displaying 6 degrees of freedom animation object described in Collada (COLLAborative Design Activity) format that defines an open standard XML (Extensible Markup Language) schema for exchanging digital assets among applications that include graphics. Without Collada standard those applications would store their graphics related resources in formats incompatible with each other, causing distribution of those assets more problematic. Collada format has been used in this project to consolidate flight data gathered by MPU-6000 chip and processed in PIC24FJ256GB106 processor, with its 3D spatial representation in Android application.

❖ Implemented transformations convert axes representations and coordinate systems.

❖ Examples of Android devices:

- freescale i.MX53 board with 3D graphics developed using Inflexion UI by Mentor Graphics with Inflexion Engine loaded onto Android OS, touch screen LCD and BSP (Board Support Package) by Adeneo Embedded [17] – used for this project, shown in Fig. 16.

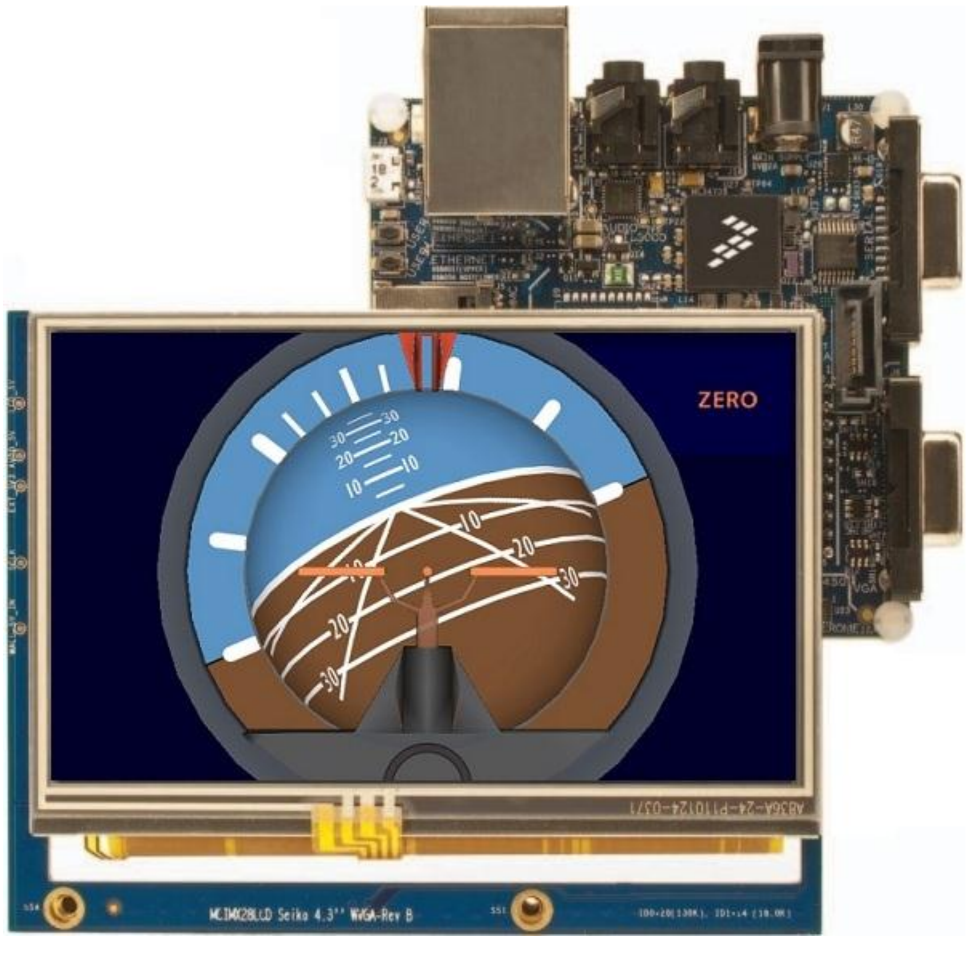


Fig. 16) Attitude Indicator App running on i.MX53 development board with IMX28LCD touch screen

- Off the shelf Android smart phones or tablets with Inflexion Engine preloaded (usually by OEM)

- Off the shelf Android smart phones or tablets with graphics developed using OpenGL library

❖ Complete Aircraft Artificial Horizon Solution

- Motion Processing Unit (MPU-6000) with integrated Accelerometer and Gyroscope, hardware DMP (Digital Motion Processor) with hardware accelerator engine and a secondary I2C-master port that interfaces to optional 3rd party digital accelerometers, and FIFO to store MotionFusion data

- Application Processor with Motion-Fusion algorithms implemented on it.

The generic Motion-Fusion library available on Invensense website [6] in fall 2011 has been modified to handle PIC24FJ256GB106 processor for the purpose of this project. Some of the functionalities that came with the library worked but most vital ones turned out not to. Due to the fact that Invensense's IP runs in encrypted form and is considered proprietary and confidential, it turned out too difficult to find the fix for encountered problems due to very limited debugging access and the proprietary library did not work “out-of-the-box as-is”. The DMP (Digital Motion Processor) memory interface uses three registers on the IMU/MPU device. “The 16-bit memory address is selected by two 8-bit registers (DMP Bank and DMP Start Address), and a third register (DMP Read/Write) provides sequential read/write access to the DMP memory using sequential I^2C read/write. The implementations provided for AT32 break the memAddr argument into Bank and Start Address, and perform single-byte writes to the Bank and Start Address registers." [19] Those registers are undocumented as they constitute the part of Invensense IP.

Few attempts were made on the Invensense's development forum and through contacting Invensense directly to resolve the issues, with no success. Eventually, Invensense removed a generic library from their website. The only library available at the time of writing this report is the one that handles Invensense's demo board with Atmel processor populated on it. Due to the problems encountered in the past with the generic library, the approach was changed and own motion processing algorithms were incorporated into the firmware, which seems to be more valuable from the research standpoint, but likely provides worse solution that the one by Invensense with its years of experience in the field and multimillion dollars resources. Based on Invensense history, which is "the pioneer and a global market leader in intelligent motion processing solutions that enable a motion-based user interface for consumer electronics." [18] it is assumed that the final attitude of the aircraft resulting from proprietary Motion-Fusion data processing would be more accurate than the solution devised for this project.

The most difficult part, as it turned out during the design and testing phase, was to eliminate a linear acceleration from the readings to obtain pure gravity that translates directly to the tilt of the object and therefore to its position in a 3D space. Due to the Motion-Fusion processing, a gyroscope's data are affected by the linear acceleration if the latter is not completely removed from the accelerometer output. The accelerometer is used in motion processing to minimize the effect of gyroscope's inherent drift caused by inevitable bias and the integration of the consecutive readings. The integration is used to convert an angular rate that gyroscope outputs in deg/sec into an actual tilt for each particular axis separately.

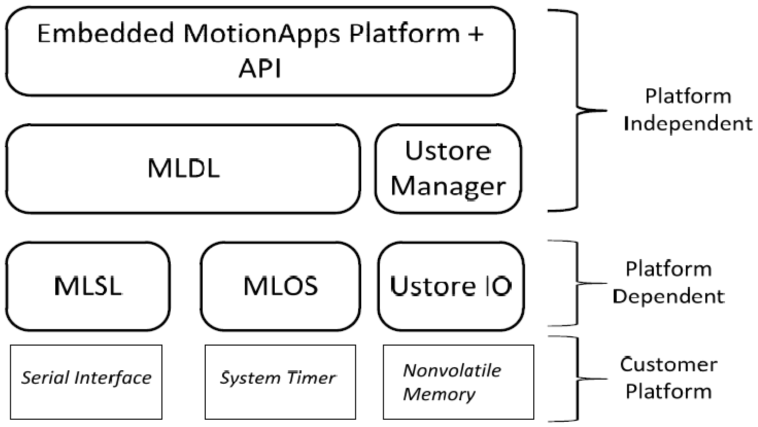


Fig. 17) Embedded MotionApps platform by Invensense [19]

The Embedded MotionApps platform, shown in Fig. 17, interfaces with the customer platform through the following modules that were prepared for Invensense IP library integration with the existing firmware [19]:

- MLSL - platform dependent I^2C implementation for communication with MPU.

- MLOS - provides a system timer with the resolution of 1ms and a delay routine.

- UStore IO - provides an access to a non-volatile memory like EEPROM, filesystem, external Flash drive, etc.

- Android device with 3D representation of one of the standard artificial horizon instruments installed in aircrafts

- MPU-6000 and Application Processor share the same PCB connected to Android device via wired USB

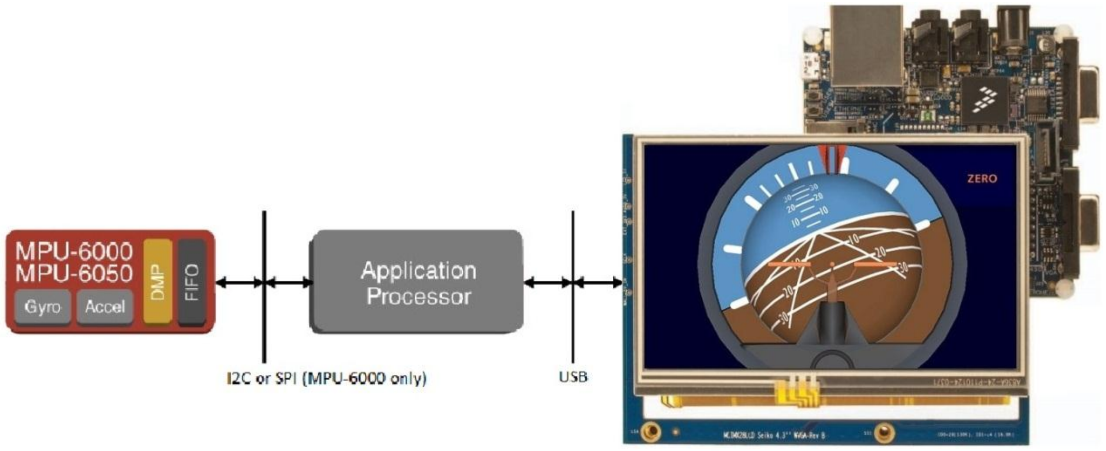


Fig. 18) Serial interfaces location in the scope of the design modules

- Application Processor provides I2C interface to MPU-6000 and USB interface to i.MX53 Android device as depicted in Fig. 18.
- Application processor performs Motion Fusion calculations using data received from MPU-6000 and provides Android device with the current position (pitch and roll) of the object, which the PCB is attached to, i.e. aircraft's cockpit

3.4. Gyroscopic chip description.

❖ Features of MPU-60X0 (Motion Processing Unit) chip by Invensense (axes orientation shown in Fig. 19):

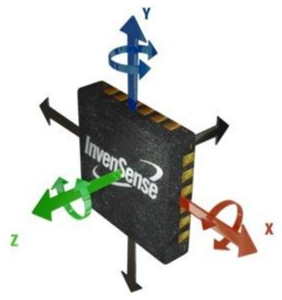


Fig. 19) MPU-60X0 axes orientation

- Complete solution to deliver 6-axis MotionFusion™ data encompassing 3-axis gyroscopes and 3-axis accelerometers

- Internal 3-axis gyroscope integrated with internal 3-axis accelerometer on one silicon die

- World's first and only integrated 6-axis IMU (Inertial Measurement Unit) to eliminate PCB (Printed Circuit Board) level cross axis misalignment errors

- Can drive 3-axis external magnetometer
- Manufactured using Nasiri process that combines MEMS (Micro-Electro-Mechanical-System) on CMOS
- Two types: MPU-6000 (I2C, SPI) and MPU-6050 (I2C, extra power options)
- Comes with 9-axis (9 degrees of freedom: 3-axis accelerator, 3-axis gyroscope, 3-axis magnetometer) proprietary Embedded MotionApps Platform™ library software (Embedded MPL), optimized for 8-bit embedded microcontrollers, capable of processing complex 9-axis MotionFusion algorithms
- 4x4x0.9mm (QFN) footprint
- VDD Supply voltage range of 2.375V–3.46V. VLOGIC (MPU-6050 only) at 1.8V±5% or VDD
- 400kHz Fast Mode I²C or up to 20MHz SPI (MPU-6000 only) serial interfaces

❖ Digital Motion Processor.

"The embedded Digital Motion Processor (DMP) is located within the MPU-60X0 and offloads computation of motion processing algorithms from the host processor. The DMP acquires data from accelerometers, gyroscopes, and additional 3rd party sensors such as magnetometers, and processes the data. The resulting data can be read from the DMP's registers, or can be buffered in a FIFO. The DMP has access to one of the MPU's external pins, which can be used for generating interrupts. The purpose of the DMP is to offload both timing requirements and processing power from the host processor. Typically, motion processing algorithms should be run at a high rate, often around 200Hz, in order to provide accurate results with low latency. This is required even if the application updates at a much lower rate; for example, a low power user interface may update as slowly as 5Hz, but the motion processing should still run at 200Hz. The DMP can be used as a tool in order to minimize power, simplify timing, simplify the software architecture, and save valuable MIPS on the host processor for use in the application." [16]

❖ MPU-6000 SERIAL INTERFACE

The MPU-6000 communicates to a system processor using either SPI or I^2C serial interface. For this project, I^2C interface has been chosen with the maximum 400kHz frequency of the SCL signal. The MPU-6000 always acts as a slave when communicating to the system processor. The LSB of the of the I2C slave address is set by the pin 9 of the chip. The logic levels for communications between the MPU-60X0 and its master are as by the VDD. [16]

3.5. Embedded implementation.

- ❖ MPU-6000 device by Invensense uses a microcontroller by Microchip (PIC24FJ64GB106) to establish and maintain USB communication channel with i.MX53 Android device. PIC24FJ64GB106 acts as a mid-man that communicates with MPU-6000 using I2C interface and protocol described in MPU-6000 specification on one end, and also it communicates with i.MX53 using USB interface and ADB (Android Debug Bridge) protocol on the other end. Other functions of PIC24FJ64GB106 microcontroller include setting up MPU-6000 at start-up, responding to MPU-6000 messages by taking a proper action, storing and processing readings coming from MPU-6000, as well as performing motion-fusion calculations resulting in actual object's position in relation to earth.

- ❖ Schematic of the circuit, shown in Fig. 20.

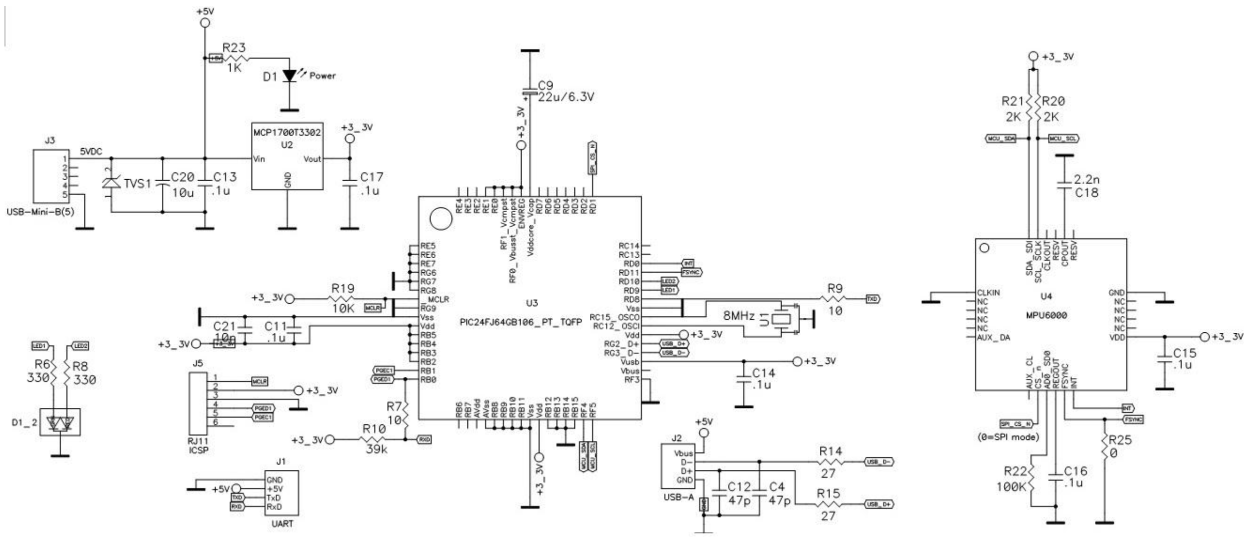


Fig. 20) Embedded circuit schematic

- J1 – UART port used strictly for diagnostics
- J2 – USB communication port to connect to Android device
- J3 - +5VDC power input
- J5 – ICSP programming connector to download/upload firmware to/from the microcontroller
- D1 – power indicator LED
- LED1,2 – auxiliary LEDs

❖ PCB (Printed Circuit Board) depicted in Fig. 21-22.

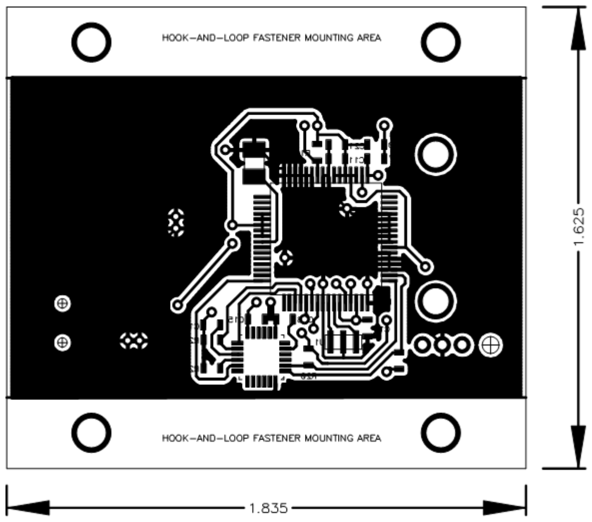

HOOK–AND–LOOP FASTENER MOUNTING AREA
1.625
HOOK–AND–LOOP FASTENER MOUNTING AREA
1.835

Fig. 21) Embedded circuit PCB (bottom side)

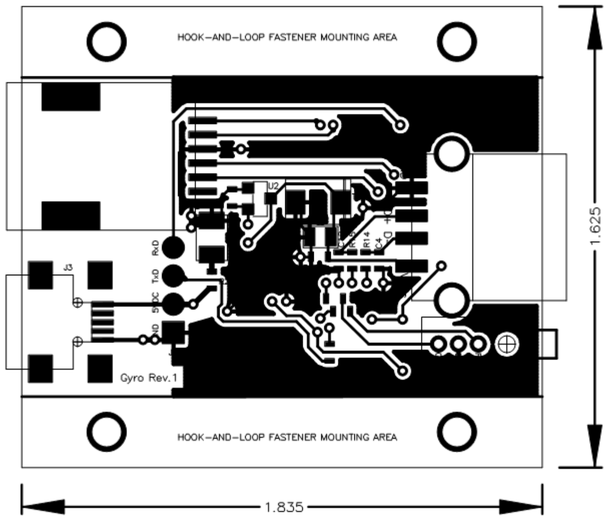

HOOK–AND–LOOP FASTENER MOUNTING AREA
U2
J3
RxD
TxD
5VDC
GND
Gyro Rev.1
1.625
HOOK–AND–LOOP FASTENER MOUNTING AREA
1.835

Fig. 22) Embedded circuit PCB (top side)

❖ Bill of Materials is provided in Appendix C to this report.

❖ PIC24FJ64GB106 microcontroller simplified specification [10]

- CPU
  - Up to 16 MIPS performance
  - 16 x 16 Hardware Multiply, Single Cycle Execution

    - 12-bit x 16-bit Hardware Divider
    - C Compiler Optimized Instruction Set

- Flash Program Memory

    - Self-Reprogrammable under Software Control
    - 10,000 erase/write cycles
    - 20 year data retention

- System

    - Internal oscillator support - 31 kHz to 8 MHz, up to 32 MHz with 4X PLL
    - On-chip LDO Voltage Regulator
    - JTAG Boundary Scan and Flash Memory Program Support
    - Fail-Safe Clock Monitor – allows safe shutdown if clock fails
    - Watchdog Timer with separate RC oscillator

- Universal Serial Bus Features

    - USB v2.0 On-the-Go compliant
    - Dual role capable, can act as either Host or Device
    - Low speed(1.5Mb/s) and full speed(12 Mb/s) operation in host mode
    - Full speed USB operation in Device mode
    - Supports 32 endpoints
    - On-chip USB transceiver

- Peripherals

    - CTMU supports Capacitive Touch applications
    - Peripheral Pin Select allows I/O remapping of many peripherals in real time
    - 4xUART Modules with LIN and IrDA support, 4 Deep FIFO
    - 3xSPI ™ Modules with 8 Deep FIFO
    - 3xI2C™ Modules with Master and Slave Modes
    - Five 16-bit Timer Modules
    - Up to 9 Input Capture and 5 Output Compare/PWM with dedicated time base
    - Hardware RTCC, Real-Time Clock Calendar with Alarms
    - PMP, Parallel Master Port, with 16 Address Lines, and 8/16-bit Data

❖ I2C interface between PIC24FJ64GB106 and MPU-6000

- MPU-6000 uses standard I^2C communication in a Fast-mode (SCL clock frequency up to 400kHz) [11]. I^2C interface on the PIC24FJ64GB106 side has been implemented to meet MPU-6000 requirements enumerated in [16].

❖ USB interface between PIC24FJ64GB106 and i.MX53

- USB 2.0 with PIC24FJ64GB106 acting as a USB host and i.MX53 acting as a USB device.

3.6. Gyroscope/Accelerometer readings processing.

❖ Gyroscope features

- Gyro operates at high resonant frequency for better rejection of ambient noise and vibration and also provides for less sensitivity to physical shock (10,000g) compared to other solutions available on the market. [20] Gyro’s resonant frequency is associated with the fact that when it rotates around any of the sense axes, the so called Coriolis effect, which has to do with a deflection of moving object(s) when they’re viewed in a rotating frame of reference [17], causes a vibration that is detected by a capacitive pickoff. Then the resulting signal is amplified, demodulated, and filtered to produce a voltage that is proportional to the angular rate. [16]. “All InvenSense X- and Y-axis gyroscopes are based on coupled dual-mass (tuning fork) proof-masses that are driven out-ofplane and generate Coriolis forces in-plane” [22], as shown in the Fig. 23:

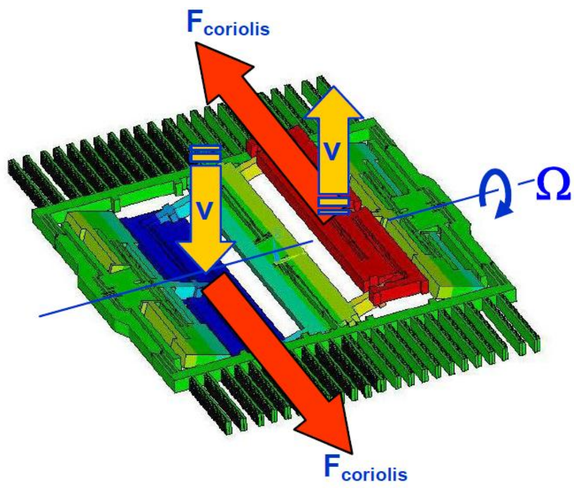


Fig. 23) X-axis gyroscope driven mode [22]

- Gyroscope measures angular speed in dps (degree per second) around three axis (x, y, z). Positive readings are obtained for counter-clockwise direction and vice versa.

- Gyroscope is specifically designed not to measure linear acceleration and to reject gravity force.

- Angle is derived by integrating rate of rotation oven certain period of time.

- Gyroscope readings always have some bias that changes over time.

- Due to integration of gyroscope readings in order to receive an actual angle, a bias translates to inevitable drift (sitting device will always drift, e.g. from 0deg to 360deg on each given axis, with different rate, depending on the integral time constant as well as initial and current bias).

- ±2000dps max (MPU-60X0).

❖ Accelerometer features:

- Accelerometer measures linear acceleration and tilt due to gravity. It can’t measure yaw, which is a rotation in relation to gravity

- Accelerometer measures acceleration in m/s2 applied to the object and after processing input data it returns three angles corresponding to its three axis (MPU-60X0).

- Angles calculated by accelerometer hardware are affected by both, linear acceleration and gravity, thus only sitting device will read pure gravity.

- In order to receive device’s attitude (pitch and roll), a linear acceleration has to be removed and only pure gravity that directly translated to actual tilt has to be taken into account

- ±16g max (MPU-60X0)

❖ Accelerometer – Gyroscope Motion Fusion, i.e. why and how to combine accelerometer with gyroscope data

- Motion-fusion purpose

  - Provides high accuracy (accelerometer) and fast response not affected by linear acceleration (gyroscope)
  - Both sensors complement each other.
  - Gyro provides turning information (angle received through integration of the readings)
  - Accelerometer provides linear acceleration and gravity combined
  - Provides accurate 6-axis interpretation of movement in space
  - Filters out accelerator’s unintended ambient movement and vibration (removes linear acceleration that affects gravity readings - tilt)
  - Gyroscope never reads zero while stationary as it should. Instead it gives accurate angular change readings in short period of time (as long as drift within that period of time is acceptable for particular application)

- Motion-fusion Implementation

  - Acceleration applied to the object, read by the sensor: a = -g - ?F / m
  - Pass accelerator readings through low pass filter to isolate gravity (If linear acceleration was constant or it was increasing or decreasing indefinitely, then its removal from accelerometer readings would not be possible, given that acceleration was not known by other means. Fortunately, long term average of acceleration that most of the objects on Earth, including aircraft are exposed to, equals zero. This means that objects usually have constant linear acceleration heading zero, while sitting, or their acceleration increases and decreases alternatively with a long term average of zero. Therefore, in order to remove a linear element from acceleration, a low pass filter can be used.)
  - Solution introduces delay that without support of gyroscope readings, would turn out not acceptable for aircraft application as in order to, in ideal situation, remove a linear acceleration from accelerometer's output additional filtration is needed and hence a delay is added to the display update time along with the filtration of the mechanical vibrations that an accelerometer is sensitive to.
  - Pass gyroscope readings through high pass filter to remove drift (Drift's short term changes happen to be close to zero, thus the influence of this element on the final angle calculated based on gyro readings can be almost eliminated by applying high pass filtration algorithm)
  - Pass, in ideal conditions error-free, pre-processed readings through complementary filter that can easily be implemented on the 8-bit platform, or mathematically complex Kalman filter if the target platform provides enough resources for its implementation that in practice provides barely noticeable advantage over much simpler to implement complementary filter

❖ Motion-Fusion theory.

- Idea behind Complementary Filter illustrated in Fig. 24.

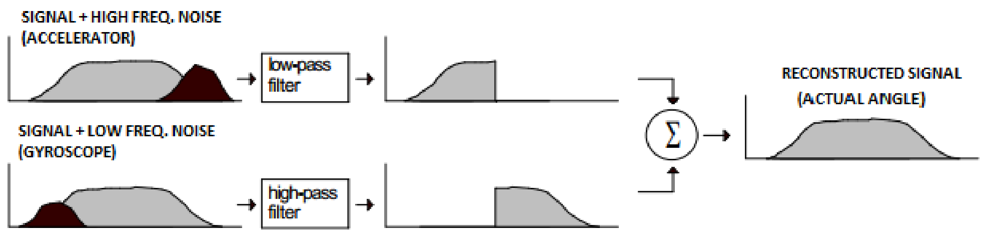


Fig. 24) Complementary filter – the concept

- Complementary filter solution decreases drift, noise, linear acceleration impact on the final angle value and lag, depicted in Fig. 25.

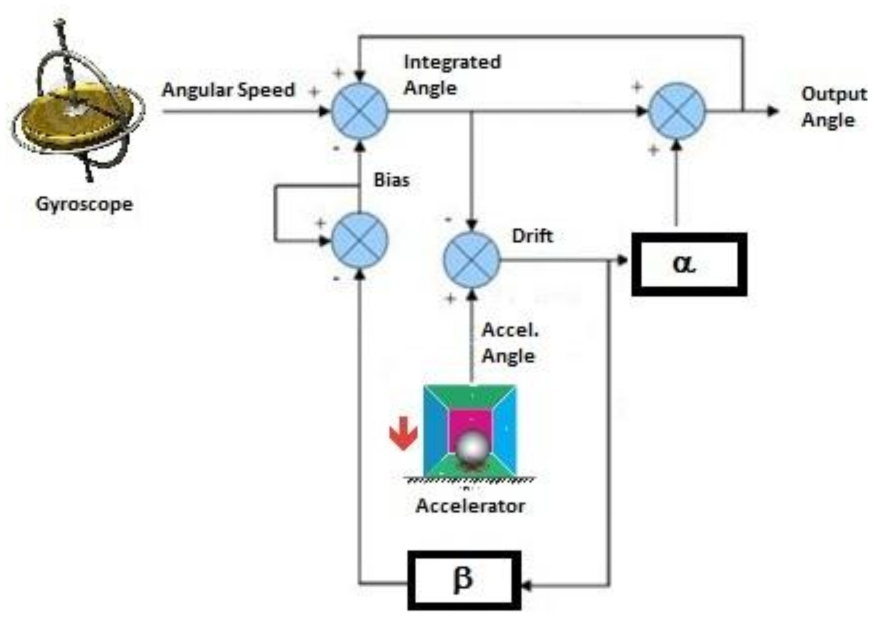


Fig. 25) Complementary filter implementation

- MPU-60X0 Motion Fusion Approach.
  - Provided by Invensense.
    - Application Processor loads a boot firmware into MPU-60X0 memory at startup
    - Application Processor performs calibration and sets up bias trackers
    - MPU-60X0 performs computations that combine data gathered from sensors and shares it with Application Processor using FIFO
    - The Algorithms for sensor fusion are InvenSense IP (run in encrypted form on the DMP)
    - User's embedded application may be affected by Invensense firmware (used as a bootloader and to set up DMP) in unacceptable way (e.g. it may affect timing requirements)
    - User is required to implement platform dependent functionalities to integrate Invensense library
    - Hardware DMP provides an object attitude via FIFO in the form of Quaternion, Rotation Matrix or Euler Angles
  - Implemented by developer.
    - Application Processor sets up MPU-60X0 using its Register Memory Map

- Application Processor reads row sensor data
- Application Processor performs Motion Fusion to combine data gathered from sensors
- User's embedded application communicates with MPU-60X0 through accessing dedicated registers
- User's embedded application does not use Invensense library
- User has to implement his own Motion Fusion algorithms, over which he has full control (they are not encrypted as Invensense IP)

▪ Motion Processing Platform as shown in Fig. 26.

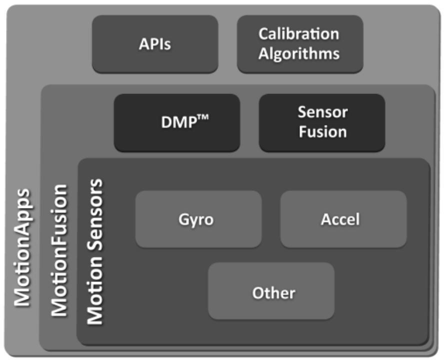


Fig. 26) Invensense Motion Processing platform

- Integrates hardware sensors, such as gyroscope and accelerometer as well as optional compass and sensors accessed through secondary serial port
- Provides computation engine to combine data gathered from individual sensors
- Provides calibration functions and API interface

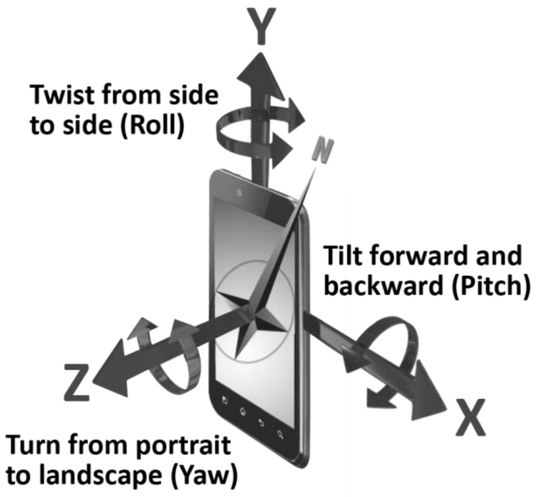

Y
Twist from side to side (Roll)
N
Tilt forward and backward (Pitch)
Z
X
Turn from portrait to landscape (Yaw)

Fig. 27) Axes orientation with reference to the Android device

- 6-axis Motion Fusion combines 3 degrees of freedom measured by Gyroscope with 3 degrees of freedom measured by Accelerator and results in angular frame relative to ground 9-axis Motion Fusion, in addition, uses 3-axis magnetometer and results in angular frame relative to both ground and north, which is illustrated in Fig. 27.

Chapter 4

Android Application

4.1. Overview of Android platform and related application development process and tools.

- Android platform

Android constitutes complete application framework built on top of Linux kernel. It is an open source software platform delivered by Google. The platform includes an Android Operating System, middleware, i.e. software that provides services to applications in addition to those ones that are already available through the operating system and Android compatible applications. Android platform has been developed as a solution to implement high-end mobile phones. A very important feature of Android platform is its open source distribution, with most of the source code provided under Apache2 licence terms, which allows proprietary modifications into the source without any source distribution requirements.

Android platform includes the stack of software components with a Linux kernel on the bottom of it. Linux kernel provides device drivers, networking handlers, security, a memory and system management functionalities. On the higher level there come libraries that support media including audio, video and 2D/3D graphics.

Dalvik VM (Virtual Machine), specifically designed for Android, implements Android runtime environment. Its main features are [23]:

- register based in contrast to stack based
- memory efficient
- uses its own byte code implementation
- provides each application with the separate copy of the VM (separate Linux process)

Application framework provides services to Android applications in the form of JAVA classes, where each and every application can share its functions with other applications and the system itself. Application layer is located above Application framework and is considered the top layer of the stack.

- Application development for Android

Eclipse IDE is used as a standard Android application development environment. It requires installation of the plugin that adds Android specific features to the integrated development environment. Application code is written in JAVA (SDK) or when using native code for Android, in C/C++ (NDK). Applications are composed of resources packaged into archives. The tools to develop Android applications are available for Linux, Windows and Mac operating systems.

Applications are described in so called “manifest”. The description of the application is used by Android system. For instance, in order to enable on hardware debugging feature, application needs to be registered as debuggable in the manifest file. [24] Applications are made up of few components. A developer uses them according to application specific requirements. Those basic components are as follows [23]:

- Activity – a functional unit of the application that may be invoked by another application, or activity.

- Service - a functional unit of the application that runs in the background, i.e. without interactive access to the user interface. It may be invoked by another application, or activity.

- Content Provider – makes data generated by one application available for other application(s) on request.

- Broadcast Receiver – the mechanism of responding to broadcast messages sent out by the system or other application(s).

Standard SDK installation includes the following development tools [25]:

- adb – Android Debug Bridge

- ddms – Dalvik Debug Monitor Service

- aapt – Android Asset Packaging tool

- dx – Dalvik Cross-Assembler

The structure of the Android application seen in the Eclipse IDE contains files as follows [26]:

- AndroidManifest.xml (required)

    - Enumerates screens provided by the application and their assignment.
    - Defines the content and data types to handle.
    - Provides information about implementation classes and cross-application information.

- src/

    - Folder stores all the source code files.

- res/ (required)

    - Folder stores all the resources used by the application, i.e. description files and external data files. Resources are compiled with the application code at build time.

- anim/
    - Folder stores animation XML files.
- layout/
    - Folder stores XML files that describe screens used by the application.
- drawable/
    - Folder stores XML or image files (.png, .jpg, .gif) to be compiled into android.graphics.drawable resources
- values/
    - classes.xml, colors.xml, strings.xml, dimens.xml, values.xml, styles.xml
- xml/
    - Folder stores XML files that can be read by the Android device at runtime
- raw/
    - Folder stores files to be copied in their original form into the target Android device

4.2. 3D graphics development.

Blender, is used to create interactive 3D graphics within Inflexion UI environment. Inflexion project is then exported as .c and .dat files that are subsequently imported through Eclipse IDE into Android application written in JAVA. Both environments communicate through variables. Inflexion framework provides JAVA functions to interact with those variables from within Android application JAVA code.

4.3. Features of Inflexion UI.

- Inflexion UI Express is an Eclipse–based software tool used to create interactive graphical user interface. The entire user interface, that might include 2D and 3D graphics, can be built using a drag and drop approach without a single line of the code to describe graphical interface itself. Graphics changes do not involve application code modifications. "Looking at the GUI, the Android development platform affords basic GUI customizations. These types of customizations include changes to boot animations, personalized wallpapers and/or icons. Swappable themes are introduced in Android, but the software developer has very little chance to create or radically customize these themes without some serious software experience. The ability to make more compelling changes, such as creating a new menu or completely changing the look and feel of a menu system, is not within the scope of the Android SDK today. Instead, these customizations need to be done at a deep code level and usually require major engineering investment." [27] Inflexion provides an intelligent, customizable GUI technology that enables the designer to create compelling interactive graphics without modifying existing code, if all shared variables that link user interface with application code are already defined and used in the Android application code. Inflexion UI location in Android OS is depicted in Fig. 28.

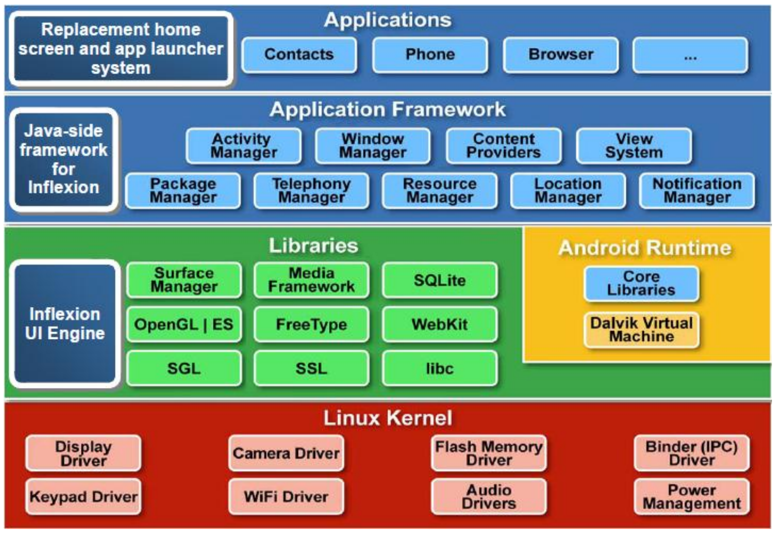


Fig. 28) Inflexion UI solution in the scope of Android platform [28]

- Graphical user interface can be functionally tested using "preview window" without updating the physical target device with design files.

- Inflexion UI consists of Inflexion Express and Inflexion Runtime. Inflexion Runtime is installed on the physical device and constitutes the engine that runs on the target device, i.MX53 in particular. The engine (library) launches executables developed using Inflexion UI. OpenGL/ES is used as a hardware graphics engine on i.MX for 2D, 2.5D, as well as 3D effects. Runtime library supports graphical effects like twisting, flipping, tilting, spinning, and is available on Android and Linux OS.

4.4. Inflexion UI installation.

- The installation is covered in the Appendix A of this report and creates the folder "<Inflexion UI install directory> \embedded_2011_03_iMX\InflexionUI-Express-2.3\SamplePackages\Palletes". That folder includes definitions of basic components that might be used to speed up user interface development, as buttons, checkBoxes, editBoxes, sliders, spins, etc. In order to make that components' database available for UI under development, the Palette has to be imported into UI IDE, by following steps described below in Inflexion UI IDE:

- Select "File > Import > General > Existing Projects into Workspace". Click "Next", as shown in Fig. 29.

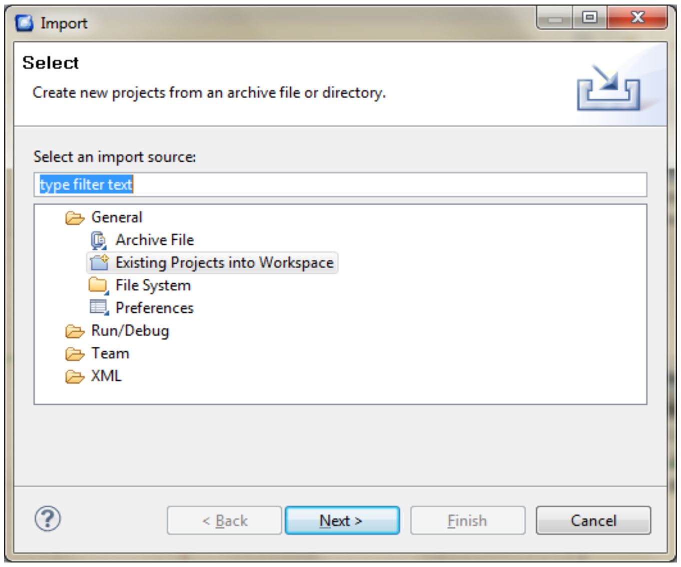
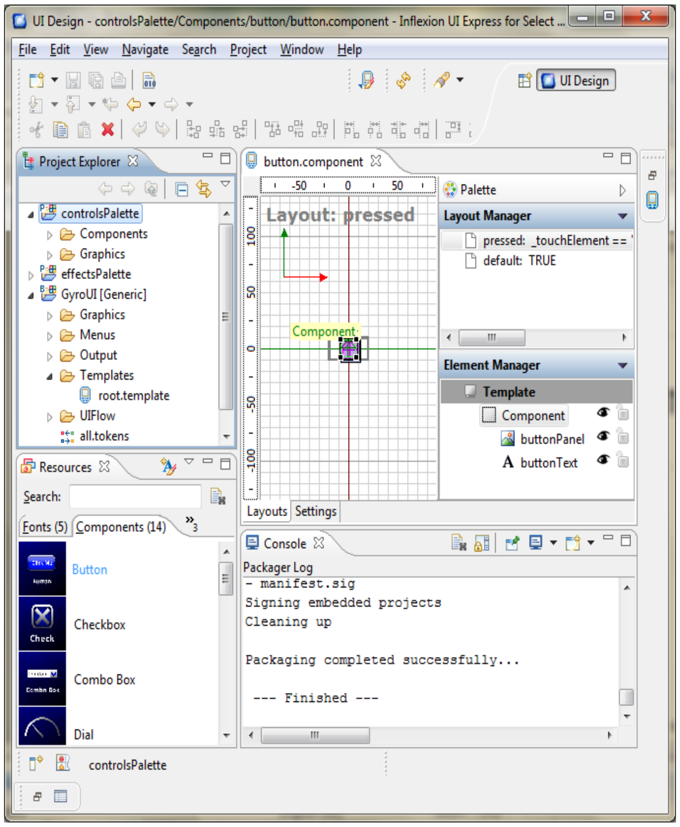


Fig. 29) Inflexion UI installation: Project import

- Browse or type "C:\mgc\embedded_2011_03_iMX\InflexionUI-Express-2.3\SamplePackages\Palettes\controlsPalette". The path may vary depending on installation settings as covered in the Appendix A of this report.

- Select active project (controlsPalette, as shown in Fig. 30) into "Import" memo box. In case there is only one item listed, simply click on "Select All". Click "Finish" to make the palette of components available for the project to develop.



```
- manifest.sig
Signing embedded projects
Cleaning up

Packaging completed successfully...

 --- Finished ---
```

Fig. 30) Inflexion UI: Selection of the active project

- In order to create graphical user interface for Android project using Inflexion UI select “New > Project... > General > Project”, enter desired name and select target location. Click “Finish”.

- Copy 3D objects saved in Collada format (COLLAborative Design Activity; files with .dae extension) created earlier in Blender and accompanying 2D graphics into “<Project directory>\GyroApp\GyroUi\Graphics”. Right click on the project's top-folder (GyroUI [Generic]) in “Project Explorer” and select “Refresh” to make 3D objects and 2D graphics immediately available within Inflexion UI environment. Objects in Collada format are used to build interactive 3D application. Fig. 31 illustrates their location.

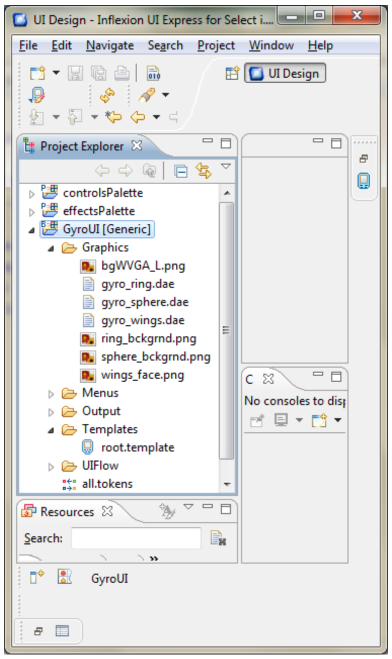


Fig. 31) Inflexion UI: Location of the Collada files

- Double click on “Project Explorer > GyroUI [Generic] > Templates > root.template” and place .dae objects and components available through Palette on the “default layout > Page” in desired locations by using “Drag & Drop” method, as shown in Fig. 32.

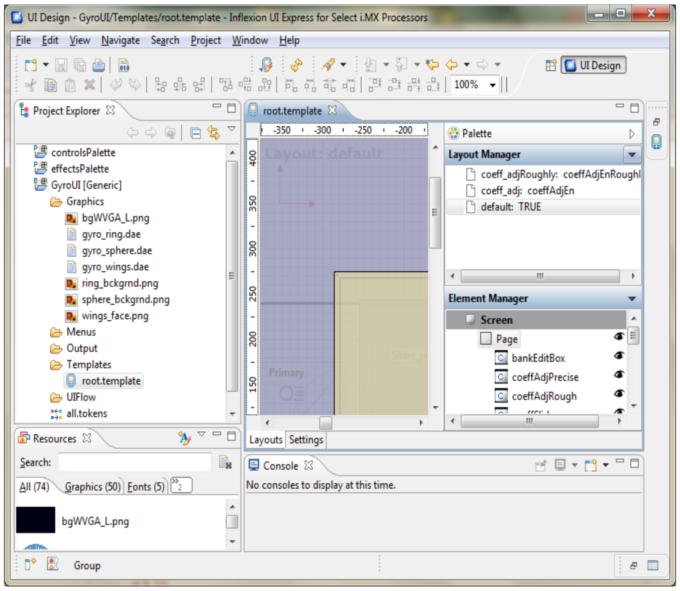
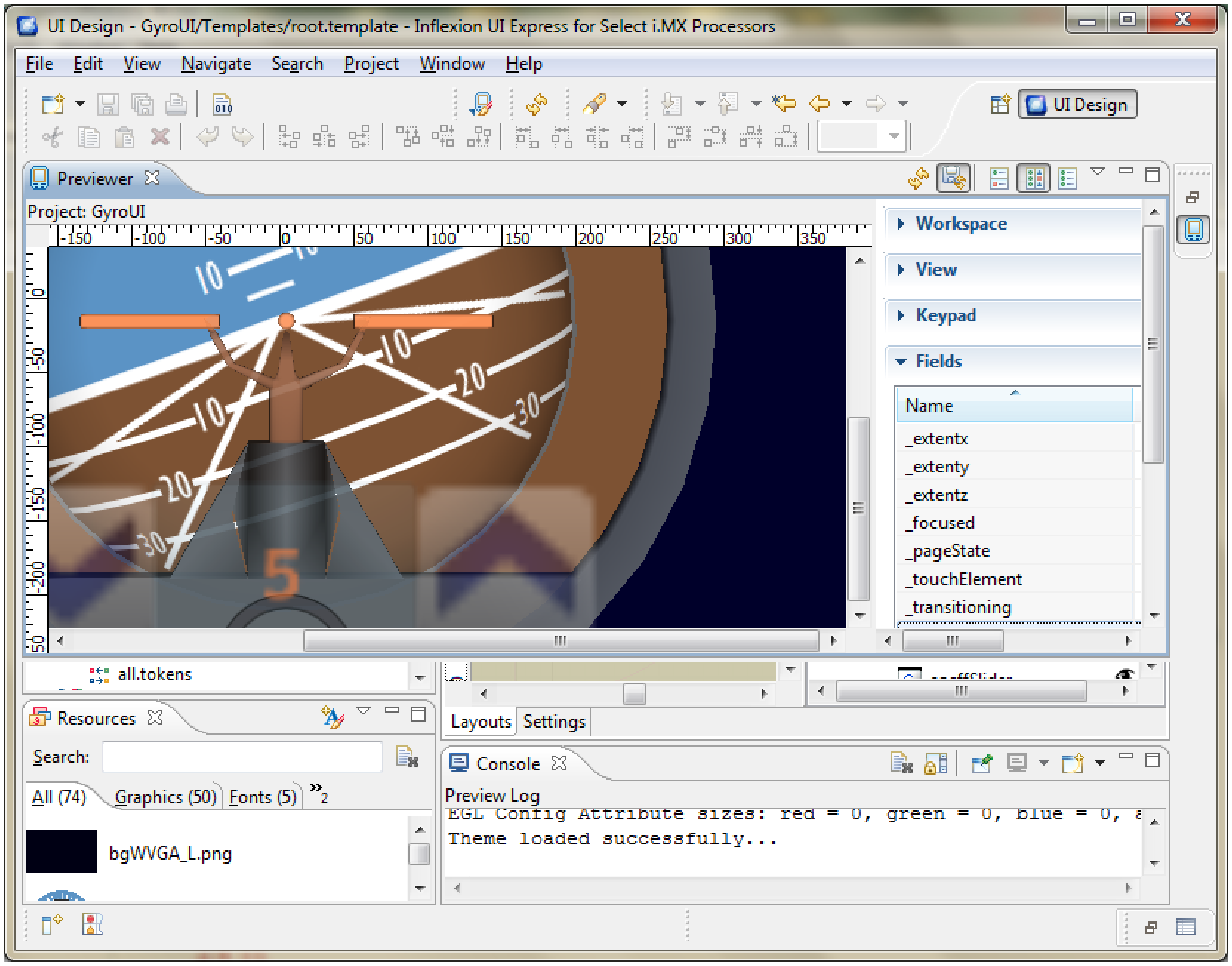


Fig. 32) Inflexion UI: The placement of the 3D objects on the layout

- At any moment, the user interface, including interactive and non-interactive graphics can be tested by right clicking the top-folder of the design, i.e. “Project

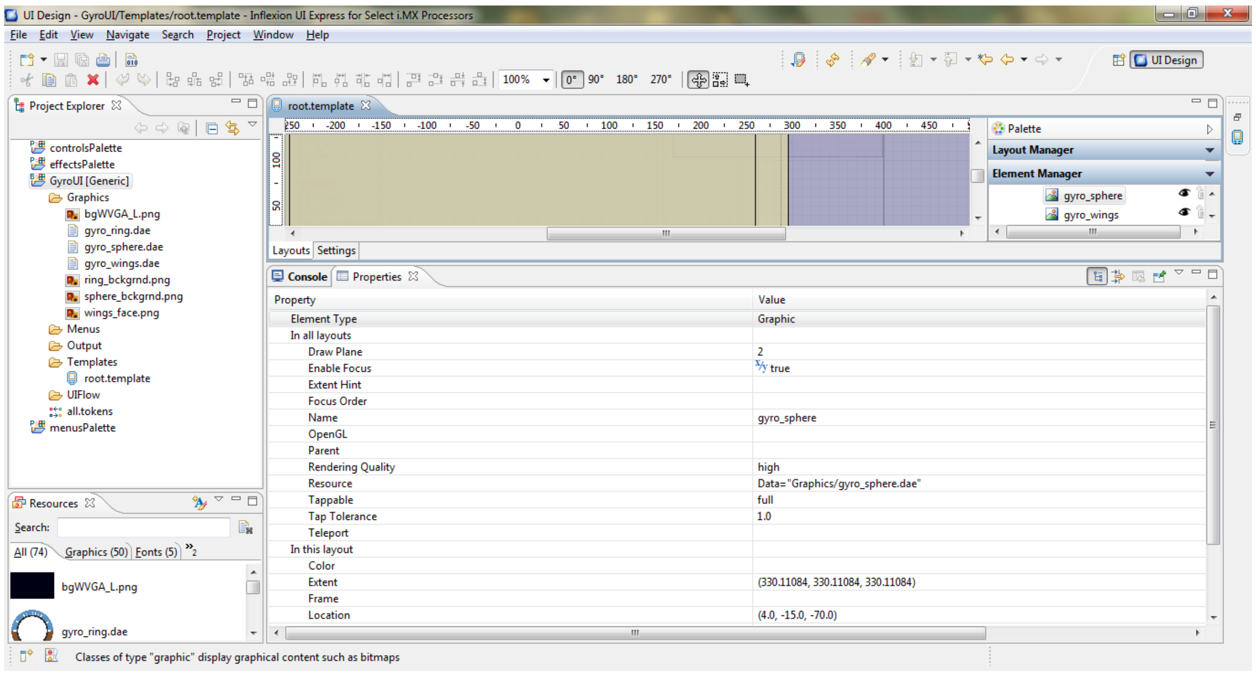
Explorer > GyroUI [Generic]" and selecting "Show in Previewer", shown in Fig. 33. The layout to preview needs to be selected first as well as its "Page" in "Element Manager > Screen" by single clicking on it. Left clicking on tested Page simulates actual tapping on Android device touch screen. "Field" provides access to variables that can be modified within UI, without integration with actual Android Application to be developed in Eclipse IDE (Android application code does not even have to exist at the moment).



```
EGL Config Attribute sizes: red = 0, green = 0, blue = 0, a
Theme loaded successfully...
```

Fig. 33) Inflexion UI: The previewer

- Access to the settings of each and every element listed by "Element Manager" is gained by right clicking on selected element, followed by left clicking on its "Properties", as shown in Fig. 34.



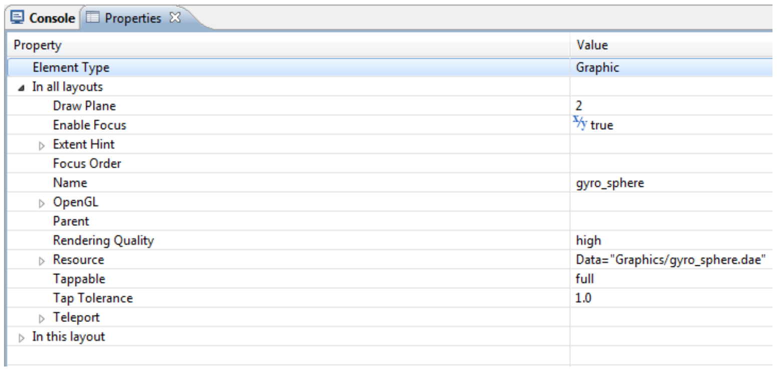
controlsPalette
effectsPalette
GyroUI [Generic]
Graphics
bgWVGA_L.png
gyro_ring.dae
gyro_sphere.dae
gyro_wings.dae
ring_bckgrnd.png
sphere_bckgrnd.png
wings_face.png
Menus
Output
Templates
root.template
UIFlow
all.tokens
menusPalette

Property	Value
Element Type	Graphic
In all layouts	
Draw Plane	2
Enable Focus	true
Extent Hint	
Focus Order	
Name	gyro_sphere
OpenGL	
Parent	
Rendering Quality	high
Resource	Data="Graphics/gyro_sphere.dae"
Tappable	full
Tap Tolerance	1.0
Teleport	
In this layout	
Color	
Extent	(330.11084, 330.11084, 330.11084)
Frame	
Location	(4.0, -15.0, -70.0)

Classes of type "graphic" display graphical content such as bitmaps

Fig. 34) Inflexion UI: Access to element's properties

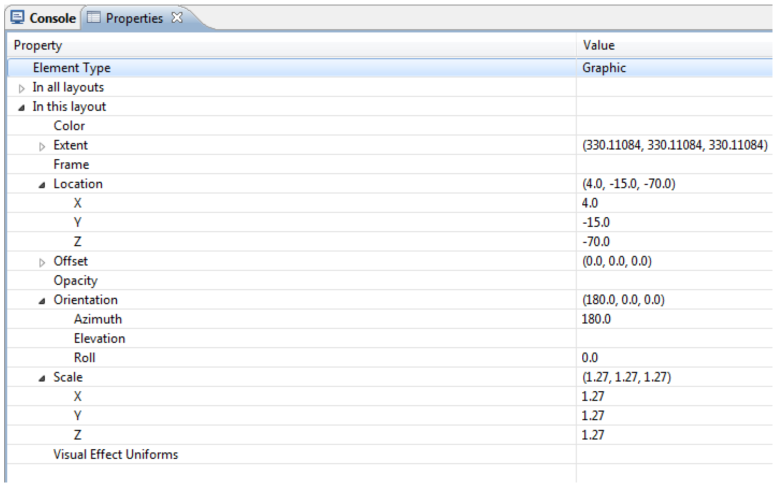
- "Draw Plane" value, shown in Fig. 35, defines the layer of the layout on which the particular object will be located, where 0 corresponds to the bottom layer, i.e. the layer that will be covered by layers with higher "Draw Plane" values. This property can be used to make object(s) with higher "Draw Plane" values visible after being placed on the object(s) with lower "Draw Plane" value(s).



Property	Value
Element Type	Graphic
In all layouts	
Draw Plane	2
Enable Focus	true
Extent Hint	
Focus Order	
Name	gyro_sphere
OpenGL	
Parent	
Rendering Quality	high
Resource	Data="Graphics/gyro_sphere.dae"
Tappable	full
Tap Tolerance	1.0
Teleport	
In this layout	

Fig. 35) Inflexion UI: Setting a draw plane

- For each layout, the location, scale in relation to original object size, and 3D orientation of the object needs to be defined, based on application dependent requirements set. Basic properties are shown in Fig. 36. Opacity, ranges from 0.0 to 1.0, and if defined, makes given object completely transparent for value = 0.0 and vice versa.

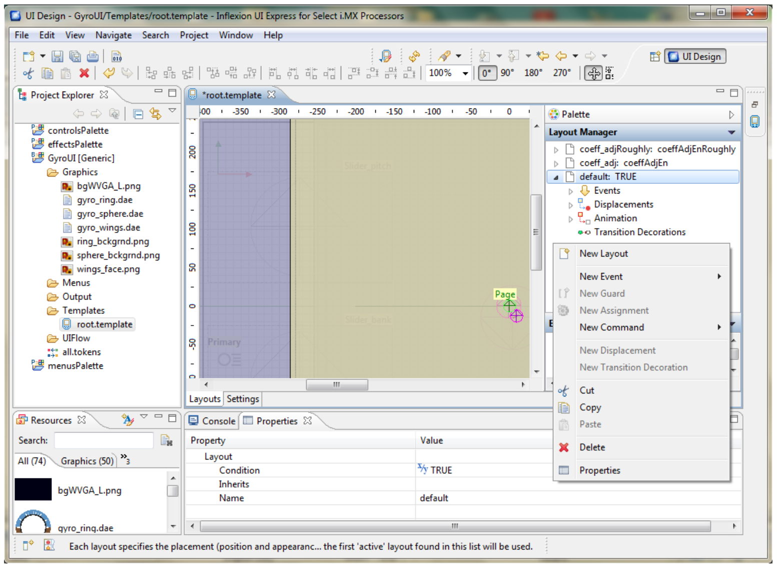


Property	Value
Element Type	Graphic
In all layouts	
In this layout	
Color	
Extent	(330.11084, 330.11084, 330.11084)
Frame	
Location	(4.0, -15.0, -70.0)
X	4.0
Y	-15.0
Z	-70.0
Offset	(0.0, 0.0, 0.0)
Opacity	
Orientation	(180.0, 0.0, 0.0)
Azimuth	180.0
Elevation	
Roll	0.0
Scale	(1.27, 1.27, 1.27)
X	1.27
Y	1.27
Z	1.27
Visual Effect Uniforms	

Fig. 36) Inflexion UI: Basic properties of the 3D object

- Inflexion UI Express uses layouts to manage and organize particular layers of interactive graphics. Layouts may inherit properties from their parent layouts, but also, they might consist of completely independent design. At any time, some layouts, that user interface is composed of, may be disabled (not available for the user) and some of them may be enabled (available and fully or partially visible, depending on whether or not the part of a given layout is covered by another layout with graphics placed on higher layers, i.e. with higher priority of visibility).

  - In order to create a new layout, right click within the "Layout Manager" area and select "New Layout" as shown in Fig. 37.

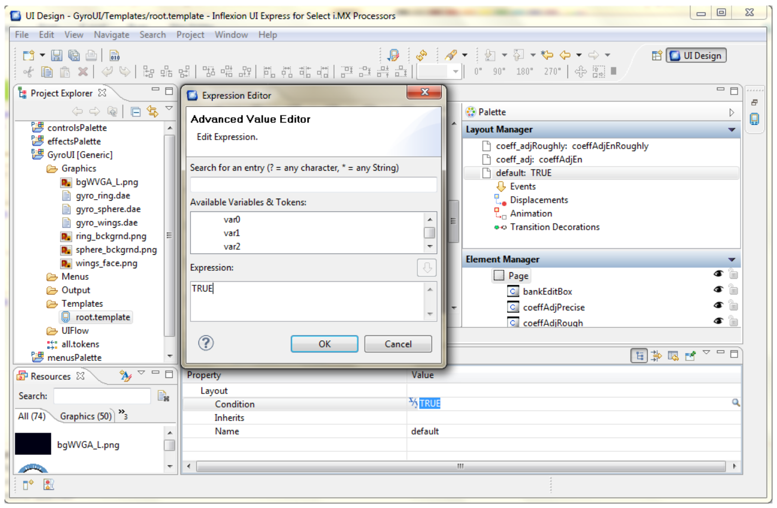


Property	Value
Layout	
Condition	TRUE
Inherits	
Name	default

Each layout specifies the placement (position and appearanc... the first 'active' layout found in this list will be used.

Fig. 37) Inflexion UI: The layout

  - Layout's "Condition", shown in Fig. 38, defines when the layout becomes available for the user. Value of “TRUE” implies the layout available unconditionally. An expression to define layout turn on/off condition may be defined by accessing layout condition editor through clicking on the magnifier icon on the right of the condition's value. An expression field accepts C statements that return boolean values, e.g. (var0 && var1).



Expression: TRUE

coeff_adjRoughly: coeffAdjEnRoughly
coeff_adj: coeffAdjEn
default: TRUE

Property	Value
Layout	
Condition	TRUE
Inherits	
Name	default

Fig. 38) Inflexion UI: Layout’s condition

- In case the layout is supposed to inherit the properties of another layout, that layout needs to be selected as the parent within “Inherits” field. Child-layout consists of interactive graphics designed within the parent one that can be further modified.

- Variables can be used by Inflexion UI internally or can be shared with Android application code developed under Eclipse IDE.

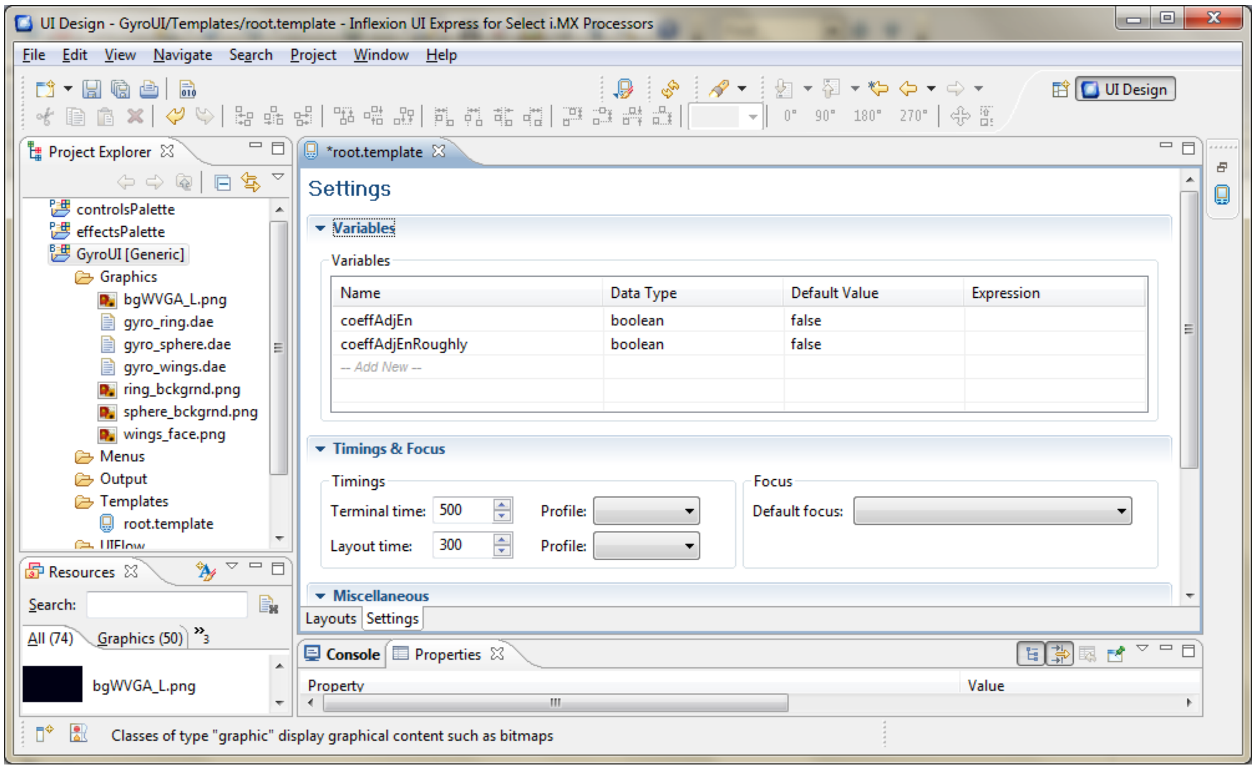
  - Internal variables – add/modify/delete access enabled through “Settings” view of the root.template. Internal Inflexion UI variables are not available for Android application code developed in Eclipse IDE in JAVA, C/C++.

    - “Name” defines the name, the particular variable is associated with

    - “Data Type”: int, string, time, float, Boolean

    - “Default Value” defines the value Inflexion UI environment assumes for a given variable without interaction with Android application.

    - “Expression” defines the constant value the variable, shown in Fig. 39, is set to



Variables

Name	Data Type	Default Value	Expression
coeffAdjEn	boolean	false	
coeffAdjEnRoughly	boolean	false	
-- Add New --			

Timings
Terminal time: 500 Profile:
Layout time: 300 Profile:

Focus
Default focus:

Fig. 39) Inflexion UI: Local variables

- Shared variables connect Inflexion UI with Android application JAVA code being developed in Eclipse IDE. Inflexion "modules" are used to link both environments through application dependent set of variables, and are defined as XML files that include "module fields" that represent particular shared variables. It's meant by "shared" that those variables can be read/modified in both, Inflexion UI IDE as well as Android application JAVA code being developed in Eclipse IDE. Those variables constitute peculiar interface between both development environments (graphics and application), through which graphical interactive user interface can be controlled based on current status of the application being executed. Shared variables can be created in the following, exemplary way:

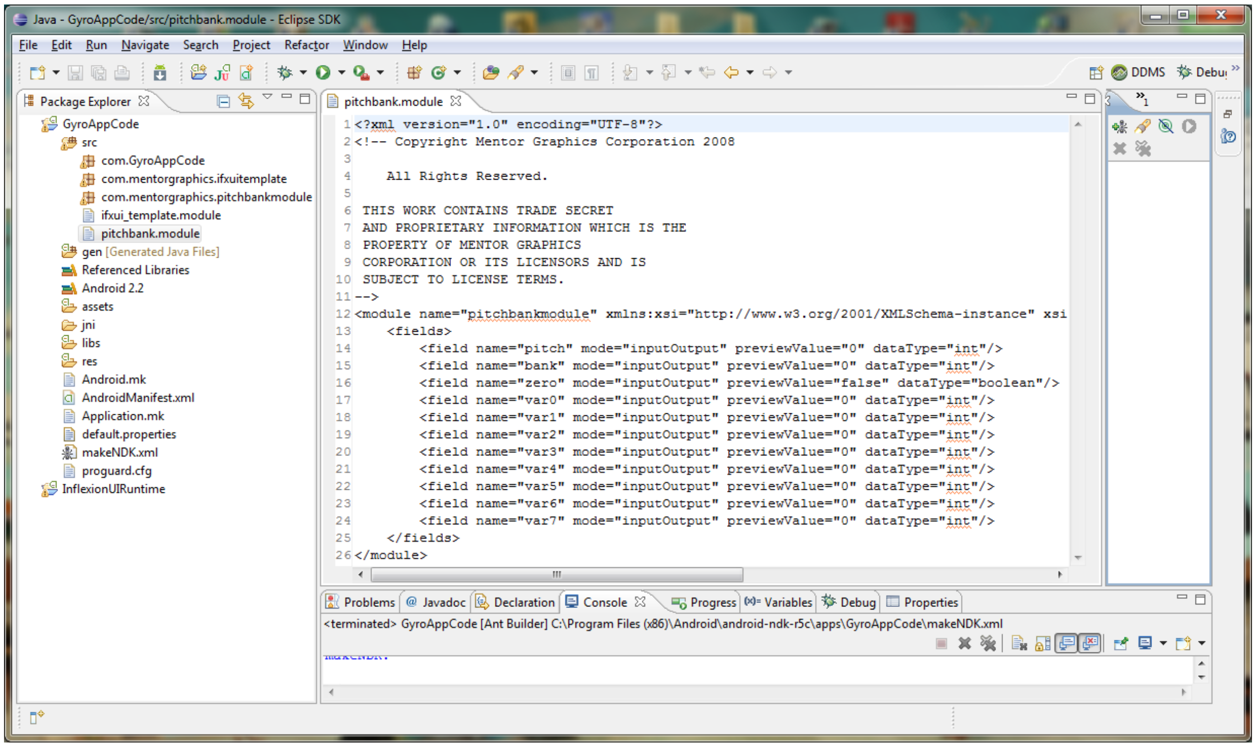
  - Create "module" XML file.

    - To simplify, copy "ifxui_template.module" from "<Eclipse> <GyroAppCode> src" and paste into the same location under different name ("pitchbank.module" will be used in this example).

    - Open created module in Inflexion UI text editor (right click and select "Open With > Text Editor") in order to modify it.

      - Enter "pitchbankmodule" as a name that Android application uses to launch the module.

- Enter required fields (one field per variable) in the following fashion, as shown in Fig. 40: “<field name="pitch" mode="inputOutput" previewValue="0" dataType="int"/>”, where “filed name” defines variable's name, “mode” defines whether the variable is to be read/written by both Inflexion UI and Android application , “previewValue” sets the value the variable assumes in Inflexion UI environment without interaction with Android application, and a “dataType” assigns one of available types (int, string, time, float, boolean) to the variable. Save the file.



```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright Mentor Graphics Corporation 2008

    All Rights Reserved.

 THIS WORK CONTAINS TRADE SECRET
 AND PROPRIETARY INFORMATION WHICH IS THE
 PROPERTY OF MENTOR GRAPHICS
 CORPORATION OR ITS LICENSORS AND IS
 SUBJECT TO LICENSE TERMS.
-->
<module name="pitchbankmodule" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi
    <fields>
        <field name="pitch" mode="inputOutput" previewValue="0" dataType="int"/>
        <field name="bank" mode="inputOutput" previewValue="0" dataType="int"/>
        <field name="zero" mode="inputOutput" previewValue="false" dataType="boolean"/>
        <field name="var0" mode="inputOutput" previewValue="0" dataType="int"/>
        <field name="var1" mode="inputOutput" previewValue="0" dataType="int"/>
        <field name="var2" mode="inputOutput" previewValue="0" dataType="int"/>
        <field name="var3" mode="inputOutput" previewValue="0" dataType="int"/>
        <field name="var4" mode="inputOutput" previewValue="0" dataType="int"/>
        <field name="var5" mode="inputOutput" previewValue="0" dataType="int"/>
        <field name="var6" mode="inputOutput" previewValue="0" dataType="int"/>
        <field name="var7" mode="inputOutput" previewValue="0" dataType="int"/>
    </fields>
</module>
```

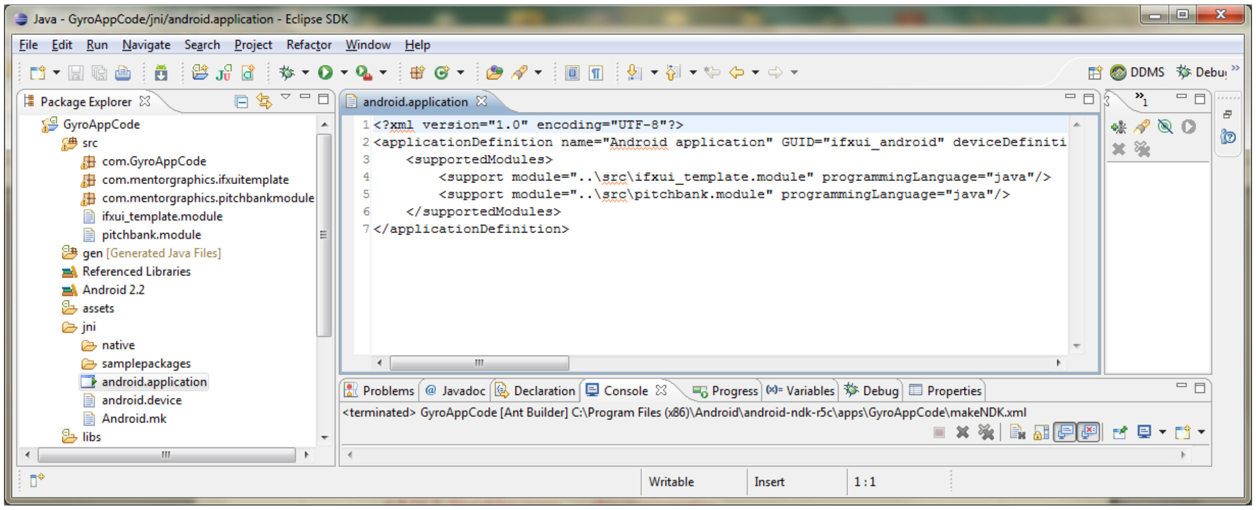
```
<terminated> GyroAppCode [Ant Builder] C:\Program Files (x86)\Android\android-ndk-r5c\apps\GyroAppCode\makeNDK.xml
```

Fig. 40) Inflexion UI: Shared variables

- Register the module in Android application so that Inflexion UI can access it, by modifying “android.application” file as follows:

    - Navigate to “<Eclipse> Package Explorer > <GyroAppCode> jni > android.application” and open the file in Text Editor available from Eclipse IDE.

    - Add “<support module="..\src\pitchbank.module" programmingLanguage="java"/>” line and save the file, to create a link between a “module” file and Android application as well as to make the module available for interactive graphical interface being developed in Inflexion UI Express. Fig. 41 depicts the registration module.



```
<?xml version="1.0" encoding="UTF-8"?>
<applicationDefinition name="Android application" GUID="ifxui_android" deviceDefiniti
    <supportedModules>
        <support module="..\src\ifxui_template.module" programmingLanguage="java"/>
        <support module="..\src\pitchbank.module" programmingLanguage="java"/>
    </supportedModules>
</applicationDefinition>
```

```
<terminated> GyroAppCode [Ant Builder] C:\Program Files (x86)\Android\android-ndk-r5c\apps\GyroAppCode\makeNDK.xml
```

Fig. 41) Inflexion UI: Module registration

- Create JAVA interface class that contains prototypes of functions used to read/write to variables shared between Inflexion UI environment and Android application being developed in Eclipse IDE and defined as fields of the module(s) file(s). JAVA interface class is created automatically every time the project is built. To build the project, navigate to “Project > Clean...” in Eclipse, select the project to build and click “OK”. Generated PitchbankmoduleInterface.java interface class will be located @ “<Eclipse> <GyroAppCode> src > com.mentorgraphics.pitchbankmodule >“.

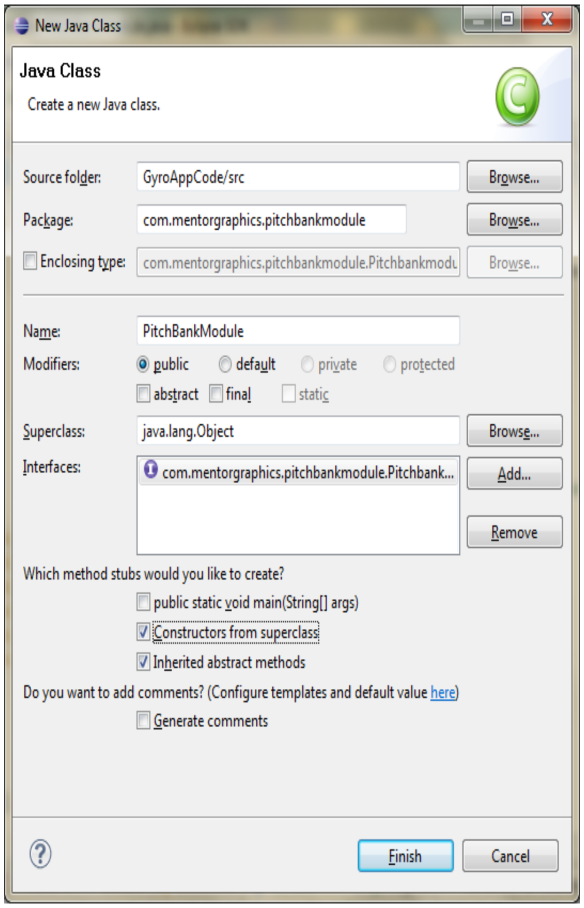
“
```
/***************************************************************************
*
*              Copyright 2006 Mentor Graphics Corporation
*                        All Rights Reserved.
*
* THIS WORK CONTAINS TRADE SECRET AND PROPRIETARY INFORMATION WHICH IS
* THE PROPERTY OF MENTOR GRAPHICS CORPORATION OR ITS LICENSORS AND IS
* SUBJECT TO LICENSE TERMS.
*
****************************************************************************/
```

```
/*****************************************************************************
*
* WARNING: This file is automatically generated. Any changes made to this
*          file may be lost.
*
*****************************************************************************/

package com.mentorgraphics.pitchbankmodule;

import java.nio.ByteBuffer;

import com.mentorgraphics.inflexionui.modules.Handle;
import com.mentorgraphics.inflexionui.modules.Module;
import com.mentorgraphics.inflexionui.modules.IfxModule;

public interface PitchbankmoduleInterface extends IfxModule {

    /* Link enums */
    public abstract int initialize(
            int                 hModuleId);

    public abstract int shutDown();

    public abstract int getFieldIntData_pitch(
            Handle<Integer>     pData);
    public abstract int setFieldIntData_pitch(
            int                 value,
            int                 isFinal);

    ...remaining fields follow:
    public abstract int getFieldIntData_nextVariableName(
            Handle<Integer>     pData);
    public abstract int setFieldIntData_nextVariableName(
            int                 value,
            int                 isFinal);
    .
    .
    ...
}
“
```

- Create module's JAVA implementation class “Pitchbankmodule.java” based on JAVA interface file already created automatically. This file will be further modified manually to meet application dependent requirements and won't be affected by subsequent project's builds as JAVA interface file will be.

"Pitchbankmodule.java" contains part of actual Android application code. To create the file follow the steps below:

- Right click "PitchbankmoduleInterface.java" and select "New > Class"
- Set the values as shown in Fig. 42, and click "Finish":



New Java Class

Java Class

Create a new Java class.

Source folder: GyroAppCode/src

Package: com.mentorgraphics.pitchbankmodule

☐ Enclosing type: com.mentorgraphics.pitchbankmodule.Pitchbankmodu

Name: PitchBankModule

Modifiers: (● public) ○ default ○ private ○ protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces: com.mentorgraphics.pitchbankmodule.Pitchbank...

Which method stubs would you like to create?

☐ public static void main(String[] args)

☑ Constructors from superclass

☑ Inherited abstract methods

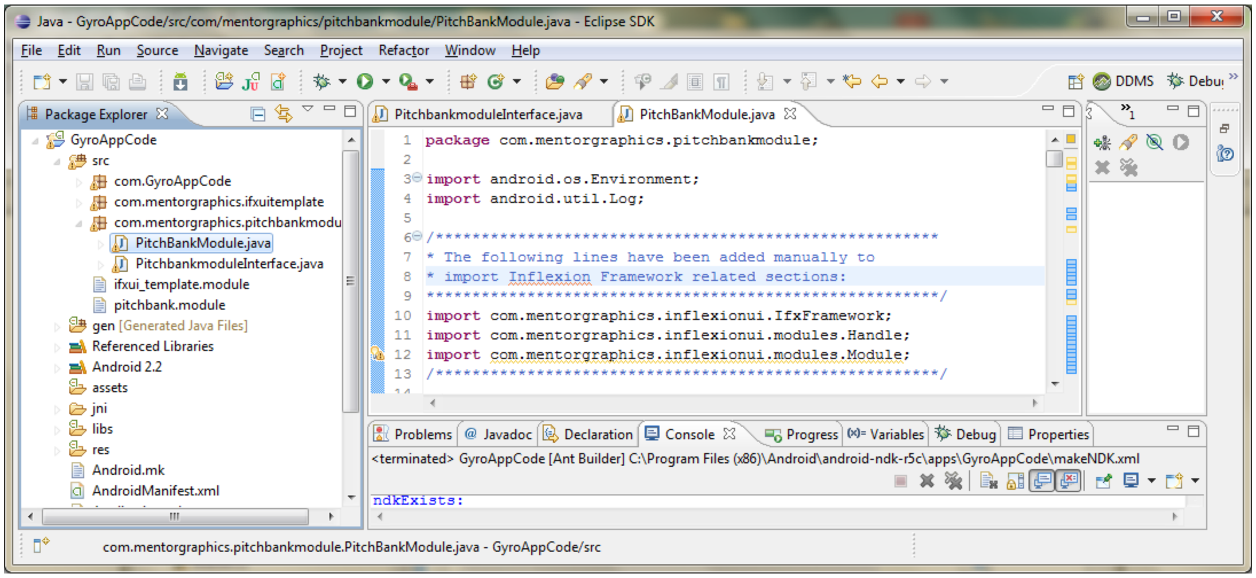
Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

Fig. 42) Inflexion UI: Java class

- Modify "PitchBankModule.java" to incorporate Inflexion Framework into Android JAVA code and implement the module.

Open "Pitchbankmodule.java" in JAVA editor available through Eclipse IDE and add import lines as shown in the Fig. 43 manually:

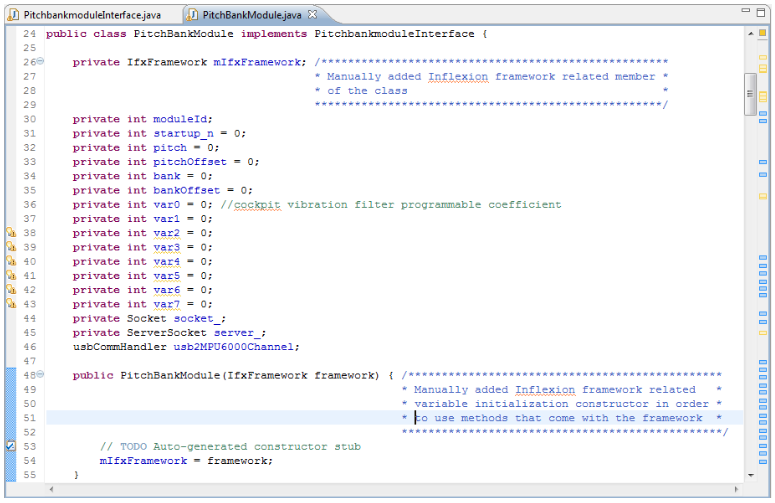


```
package com.mentorgraphics.pitchbankmodule;

import android.os.Environment;
import android.util.Log;

/*********************************************************
* The following lines have been added manually to
* import Inflexion Framework related sections:
*********************************************************/
import com.mentorgraphics.inflexionui.IfxFramework;
import com.mentorgraphics.inflexionui.modules.Handle;
import com.mentorgraphics.inflexionui.modules.Module;
/*********************************************************/
```

```
<terminated> GyroAppCode [Ant Builder] C:\Program Files (x86)\Android\android-ndk-r5c\apps\GyroAppCode\makeNDK.xml
ndkExists:
```

Fig. 43) Inflexion UI: Inflexion framework imports

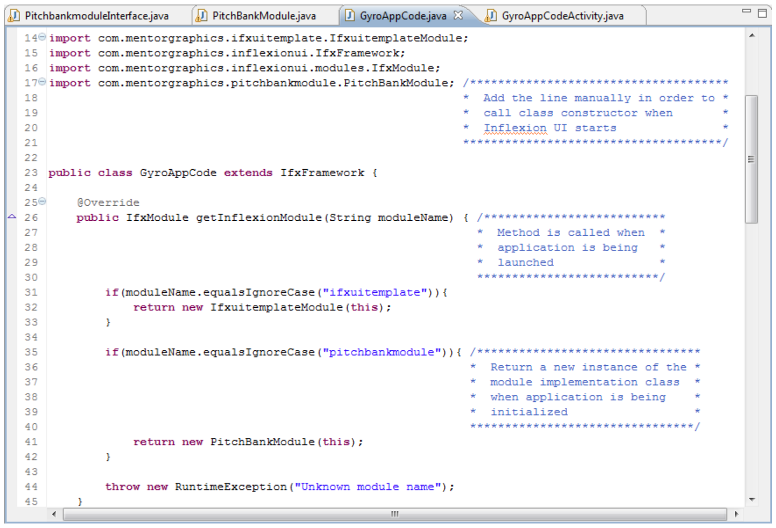
Add manually the member of the class as well as its initialization constructor as shown in the Fig. 44, then save the file:



```
public class PitchBankModule implements PitchbankmoduleInterface {

    private IfxFramework mIfxFramework; /****************************************************
                                         * Manually added Inflexion framework related member *
                                         * of the class                                      *
                                         ****************************************************/
    private int moduleId;
    private int startup_n = 0;
    private int pitch = 0;
    private int pitchOffset = 0;
    private int bank = 0;
    private int bankOffset = 0;
    private int var0 = 0; //cockpit vibration filter programmable coefficient
    private int var1 = 0;
    private int var2 = 0;
    private int var3 = 0;
    private int var4 = 0;
    private int var5 = 0;
    private int var6 = 0;
    private int var7 = 0;
    private Socket socket_;
    private ServerSocket server_;
    usbCommHandler usb2MPU6000Channel;

    public PitchBankModule(IfxFramework framework) { /************************************************
                                                      * Manually added Inflexion framework related   *
                                                      * variable initialization constructor in order *
                                                      * to use methods that come with the framework  *
                                                      ************************************************/
        // TODO Auto-generated constructor stub
        mIfxFramework = framework;
    }
```

Fig. 44) Inflexion UI: Class member

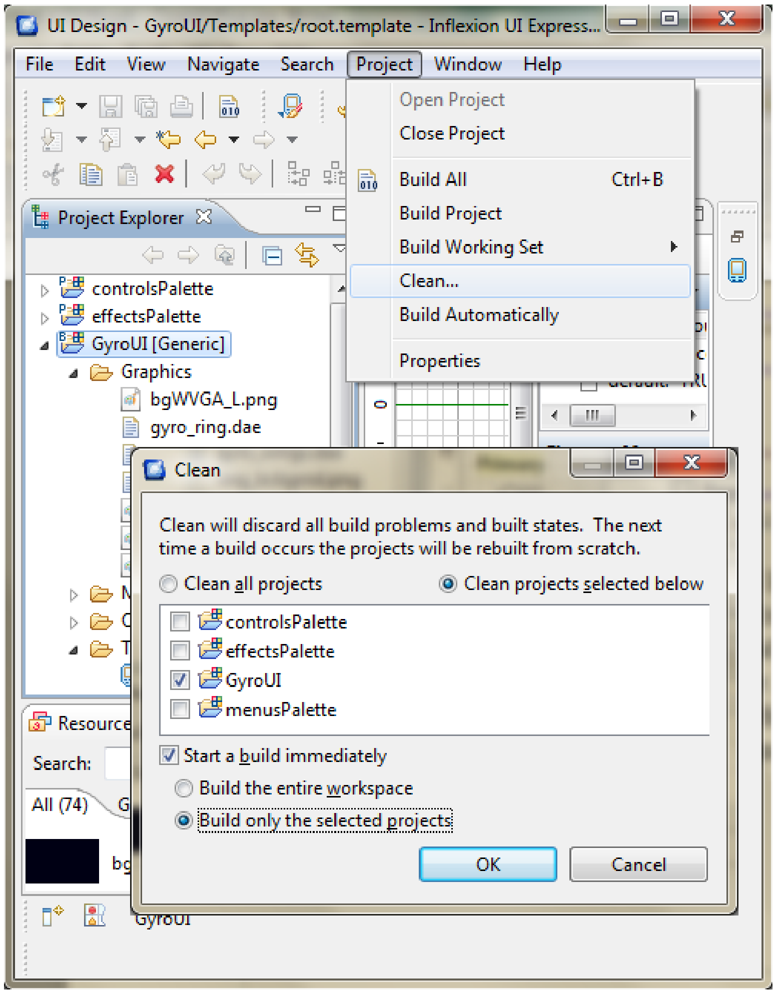
Integrate module implementation class with Android application by adding lines into “GyroAppCode.java” as shown in Fig. 45 and save the file:



```java
import com.mentorgraphics.ifxuitemplate.IfxuitemplateModule;
import com.mentorgraphics.inflexionui.IfxFramework;
import com.mentorgraphics.inflexionui.modules.IfxModule;
import com.mentorgraphics.pitchbankmodule.PitchBankModule; /*************************************
                                                            *  Add the line manually in order to *
                                                            *  call class constructor when       *
                                                            *  Inflexion UI starts               *
                                                            *************************************/

public class GyroAppCode extends IfxFramework {

    @Override
    public IfxModule getInflexionModule(String moduleName) { /*************************
                                                              *  Method is called when  *
                                                              *  application is being   *
                                                              *  launched               *
                                                              *************************/
        if(moduleName.equalsIgnoreCase("ifxuitemplate")){
            return new IfxuitemplateModule(this);
        }

        if(moduleName.equalsIgnoreCase("pitchbankmodule")){ /********************************
                                                             *  Return a new instance of the *
                                                             *  module implementation class  *
                                                             *  when application is being    *
                                                             *  initialized                  *
                                                             ********************************/
            return new PitchBankModule(this);
        }

        throw new RuntimeException("Unknown module name");
    }
```

Fig. 45) Inflexion UI: Integration of the module implementation class with Android app

- In order to build Inflexion UI project select the project to clean with "Start a build immediately" option checked as shown in Fig. 46:



Clean will discard all build problems and built states. The next time a build occurs the projects will be rebuilt from scratch.

Clean all projects [✓ Clean projects selected below]

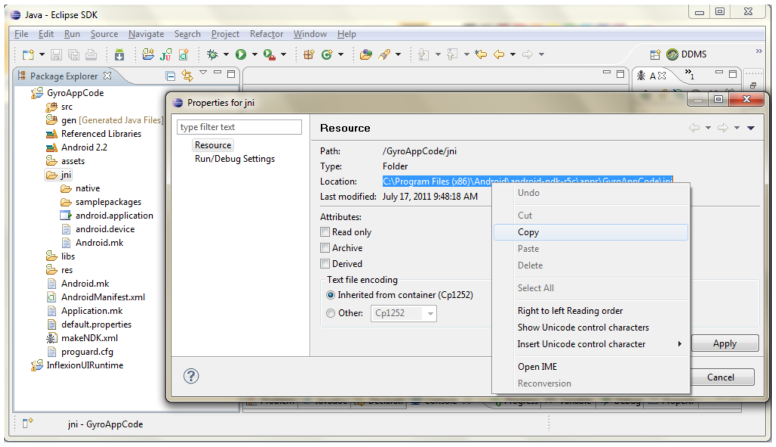
- [ ] controlsPalette
- [ ] effectsPalette
- [x] GyroUI
- [ ] menusPalette

- [x] Start a build immediately

Build the entire workspace [✓ Build only the selected projects]

Fig. 46) Inflexion UI: Project build

- Export modified Inflexion UI project into Android Eclipse IDE.
  - Link Android application project being developed under Eclipse with Interactive 3D Graphics being developed in Inflexion UI.
    - Copy the path to the "jni" folder in Eclispse IDE as shown in Fig. 47, and click "Cancel":



Resource

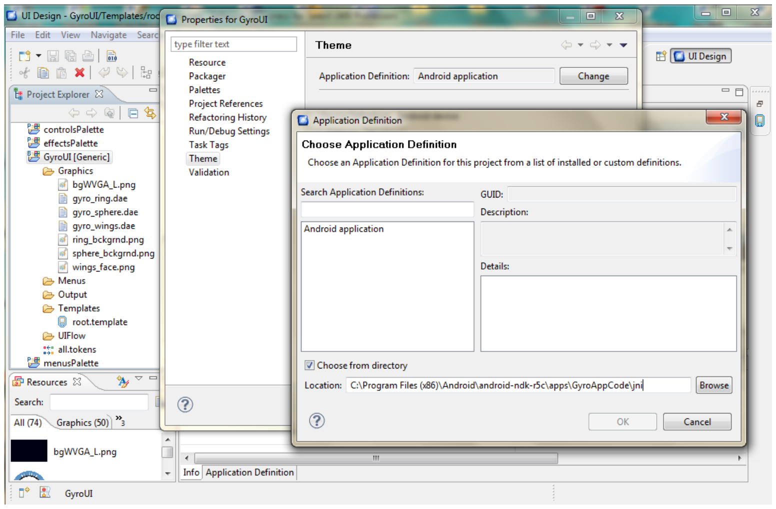
Path: /GyroAppCode/jni
Type: Folder
Location: C:\Program Files (x86)\Android\android-ndk-r5c\apps\GyroAppCode\jni
Last modified: July 17, 2011 9:48:18 AM

Attributes:
- [ ] Read only
- [ ] Archive
- [ ] Derived

Text file encoding
- (●) Inherited from container (Cp1252)
- ( ) Other: Cp1252

Fig. 47) Inflexion UI / Eclipse integration – “jni” folder in Eclipse

- Paste the path to the “jni” folder previously copied in Eclispse IDE into “Inflexion UI Express > GyroUI > Properties > Theme > Application Definition > Location” to link both environments the project is being developed in as shown in the Fig. 48:



Theme

Application Definition: Android application

Application Definition

Choose Application Definition

Choose an Application Definition for this project from a list of installed or custom definitions.

Search Application Definitions:

Android application

GUID:

Description:

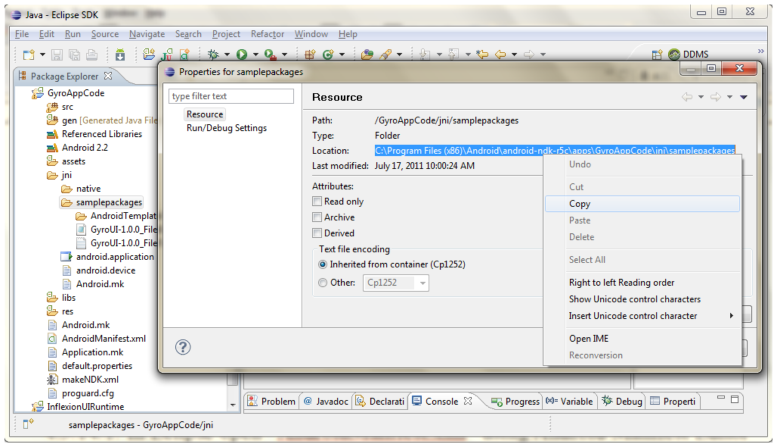
Details:

- [✓] Choose from directory

Location: C:\Program Files (x86)\Android\android-ndk-r5c\apps\GyroAppCode\jni

Fig. 48) Inflexion UI / Eclipse integration – “jni” folder in Inflexion

- Copy the path to the “samplepackages” folder in Eclispse IDE as in the Fig. 49, and click “Cancel”:



Path: /GyroAppCode/jni/samplepackages
Type: Folder
Location: C:\Program Files (x86)\Android\android-ndk-r5c\apps\GyroAppCode\jni\samplepackages
Last modified: July 17, 2011 10:00:24 AM

Fig. 49) Inflexion UI / Eclipse integration – “sample packages” folder in Eclipse

- Export .c and .dat files generated by Inflexion UI Express to Eclipse as a ROM package as depicted in Fig. 50. Interactive graphics developed under Inflexion UI becomes immediately available in Eclipse IDE and gets integrated with Android application JAVA code through the set of variables.

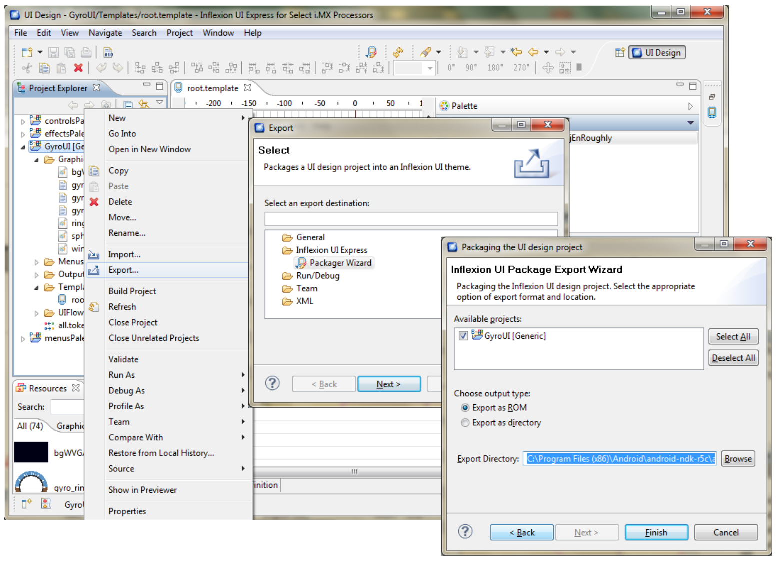

UI Design - GyroUI/Templates/root.template - Inflexion UI Express for Select i.MX Processors
Project Explorer
root.template
Palette
Export
Select
Packages a UI design project into an Inflexion UI theme.
Select an export destination:
General
Inflexion UI Express
Packager Wizard
Run/Debug
Team
XML
Packaging the UI design project
Inflexion UI Package Export Wizard
Packaging the Inflexion UI design project. Select the appropriate option of export format and location.
Available projects:
GyroUI [Generic]
Select All
Deselect All
Choose output type:
Export as ROM
Export as directory
Export Directory:
Browse
< Back
Next >
Finish
Cancel

Fig. 50) Inflexion UI: Package wizard

- Android application JAVA code interface between both development environments, Inflexion UI and JAVA in Eclipse.
  - “PitchBankModule.java” includes functions to serve as a mid-men between Eclipse and Inflexion UI environments as follows:
    - Function called by Inflexion framework to fetch the value of the variable.

In this example “pitch” is that variable:

```
public int getFieldIntData_pitch(Handle<Integer> pData) {
        pData.value = pitch_eclipse; /* pData.value represents
        “pitch” variable accessed by Inflexion UI through the
        module */
        return 0;
    }
```

- Function called by Inflexion framework whenever the variable is modified by the Inflexion UI. In this example “zero” is that variable:

```
public int setFieldBoolData_zero(boolean value) {
        pitchOffset += pitch;
        pitch = 0;
        mIfxFramework.ifxiRequestFieldRefresh(moduleId, 0, -1,
                                                    "pitch");

        bankOffset += bank;
        bank = 0;
        mIfxFramework.ifxiRequestFieldRefresh(moduleId, 0, -1,
                                                    "bank");
        return 0;
    }
```

- Function called by the Android application to inform Inflexion UI that the variable is being updated, so that Inflexion UI can call getFieldIntData_zero (for variable “zero”) to refresh the value.

```
mIfxFramework.ifxiRequestFieldRefresh(moduleId, 0, -1, "zero");
```

- Debugging.
  - In Eclipse open “AndroidManifest.xml” using Android Manifest Editor (accessible through right clicking and selecting from the list)
  - Navigate to “Application” tag
  - Set “Debuggable” option to “true”
  - Save the file
  - Set the breakpoint at desired line of desired file including JAVA code by right clicking on the line number and clicking “Toggle Breakpoint”
  - In “Eclipse > Package Explorer” right click on the project top folder and select “Debug As > Debug Configurations...”
  - Select configuration corresponding to a given project.
  - Click “Debug” button and select device currently connected via ADB (Android Debug Bridge) protocol.
  - Click “OK” in order to download and launch application on the physical

device (i.MX53).
  - Perform a proper action on the target device to cause the stop of debugging at the breakpoint.

- Downloading into Android device (running onto Android)

  - Right click on Gyro in "Eclipse IDE > Package Explorer"
  - Select "Run As > Run Configurations..."
  - Click on the icon "New Launch Configuration" if desired configuration does not exist yet.
  - Enter desired name of the configuration
  - Click "Browse" button and select the project to run
  - Navigate to "Target" tab
  - Pick "Manual" as "Deployment target selection mode"
  - Click "Run"
  - Choose a running Android device from the list that corresponds to the device under development (i.MX53)
  - Click "OK" to install and run application on the target device

Chapter 5

Development of 3D Objects and their integration with Inflexion UI

5.1. 3D Format

5.1.1. Collada (COLLAborative Design Activity) format.

Collada files describe 3D objects using XML (Extensible Markup Language) and support sharing digital resources among independent graphical applications through standardization. Collada files are extended with “.dae”, which stands for “digital asset exchange”.

5.1.2. Graphics interface between Blender suite and Inflexion UI.

Inflexion UI accepts 3D objects in Collada format (COLLAborative Design Activity) that defines interchange rules for interactive 3D applications that might include animations.

At the time of developing graphics interface for this project, Inflexion UI accepted Collada files that include features exported by Blender v. 2.49, which used, already obsolete, user interface (the last “old” version of Blender) and API (Application Programming Interface). Therefore, 3D objects were created in Blender v. 2.5, with modern GUI and API, then imported by Blender 2.49 and finally exported as .dae files to be used by Inflexion UI. Blender is a tool that utilizes OpenGL library for drawing graphics interface. It uses scripts written in Python (popular interpreted programming language), that calls on its routines in order to extend existing functionalities. “Blender is the free open source 3D content creation suite, available for all major operating systems under the GNU General Public License.” [29].

5.1.3. XML (Extensible Markup Language)

XML constitutes flexible, self-descriptive (tags are defined by developer) text format markup language, designed to handle the challenges of electronic publishing (transporting and storing data) and its interchange among independently developed applications, even on incompatible platforms.

- Features of XML [30]:
    - Used to simplify data storage and sharing.
    - Separates Data from HTML.
    - Stores data in separate XML files.
    - External XML files can be read and modified using JavaScript
    - Simplifies Data Sharing
    - XML creates a bridge between systems with data in incompatible formats. Simplifies Data Transport between incompatible systems over the Internet.
    - Stores data in plain text format, thus providing software/hardware independent mechanism of storing data.

  - Simplifies Platform Changes (data described using XML stays untouched).
  - Makes data available across different applications (HTML pages and XML data sources)
  - Used to define new Internet languages, e.g:

- A lot of new Internet languages are created with XML, e.g:

  - XHTML
  - WSDL for describing available web services
  - WAP and WML as markup languages for handheld devices
  - RSS languages for news feeds
  - RDF and OWL for describing resources and ontology
  - SMIL for describing multimedia for the web

5.2. 3D IDE.

5.2.1. Blender (Due to the complexity of the tool, only essential information related to this project has been emphasized, without providing step by step guidance. Documentation, including manuals and tutorials is available at http://www.blender.org).

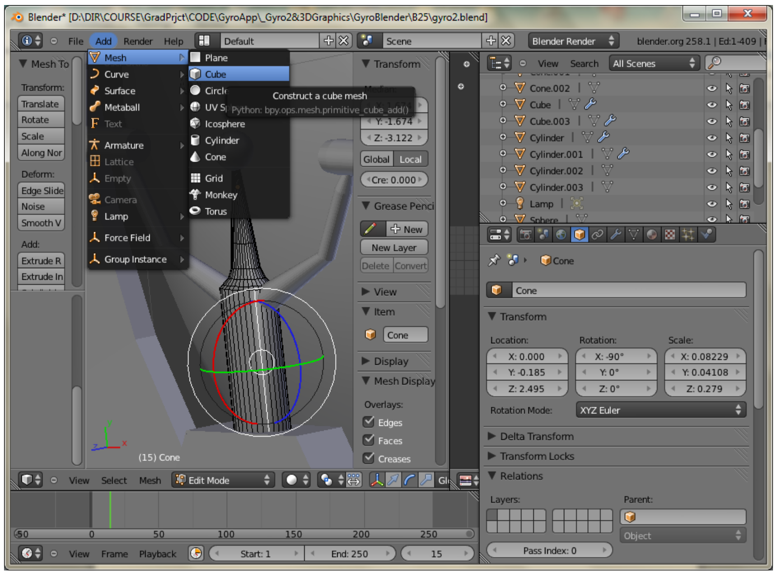
- Key features [31]:

  - Rendered and post-processed image

  - Fully integrated creation suite with broad range of essential tools for the creation of 3D content , like:

    - Modelling.

      Base objects in Blender are added to the project through menu. Depending on the application they may be further modified. Basic operations in Blender are: changing the position of the object, resizing the object and rotating the object (available through Hot-Keys or icons as follows:). Modelling in Blender is shown in Fig. 51 and basic elements of the 3D object(s) are depicted in Fig. 52.



Blender* [D:\DIR\COURSE\GradPrjct\CODE\GyroApp_Gyro2&3DGraphics\GyroBlender\B25\gyro2.blend]

Construct a cube mesh
Python: bpy.ops.mesh.primitive_cube_add()

Location: X: 0.000 Y: -0.185 Z: 2.495
Rotation: X: -90° Y: 0° Z: 0°
Scale: X: 0.08229 Y: 0.04108 Z: 0.279
Rotation Mode: XYZ Euler
Pass Index: 0
Start: 1 End: 250 15

Fig. 51) Modelling in Blender v. 2.5

3D objects consists of different elements as follows:

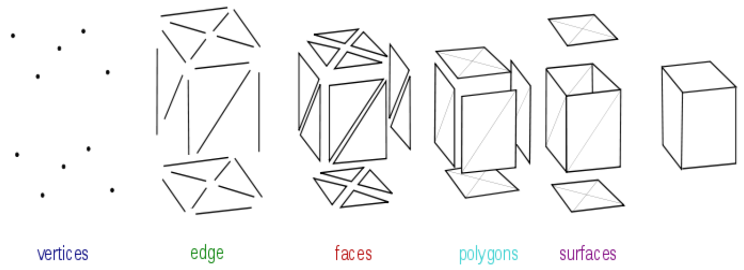


Fig. 52) Editable elements of the 3D objects [33]

Particular meshes are positioned using “Object Mode” and modified further using “Edit Mode”. Basic operations used to create desired shape include mirroring and extruding.

- Texturing, which connects triangles that make up a 3D object with the image, and UV-mapping shown in Fig. 53, defined as the process of making a 2D image representation of a 3D model. Popular nomenclature uses X, Y, Z letters to describe 3D object in the model space, and to differentiate, U, V letters to describe 2D mesh coordinates in the model space, as shown in Fig. 54.

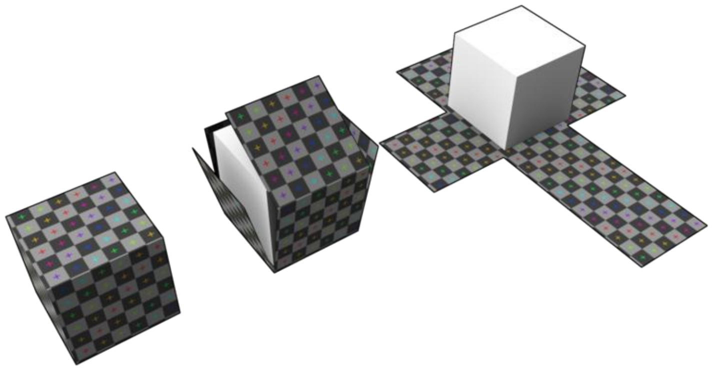


Fig. 53) UV mapping of the cube 3D object [32]

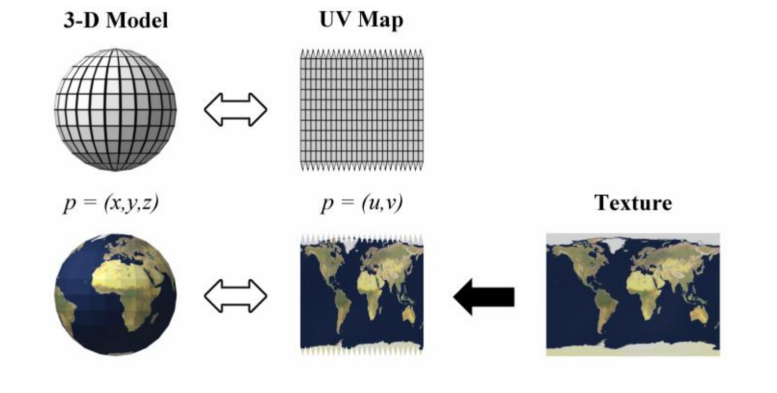


Fig. 54) 3D object texturing through UV mapping [32]

Fig. 55-58 illustrate the elements of the Attitude Indicator 3D object:

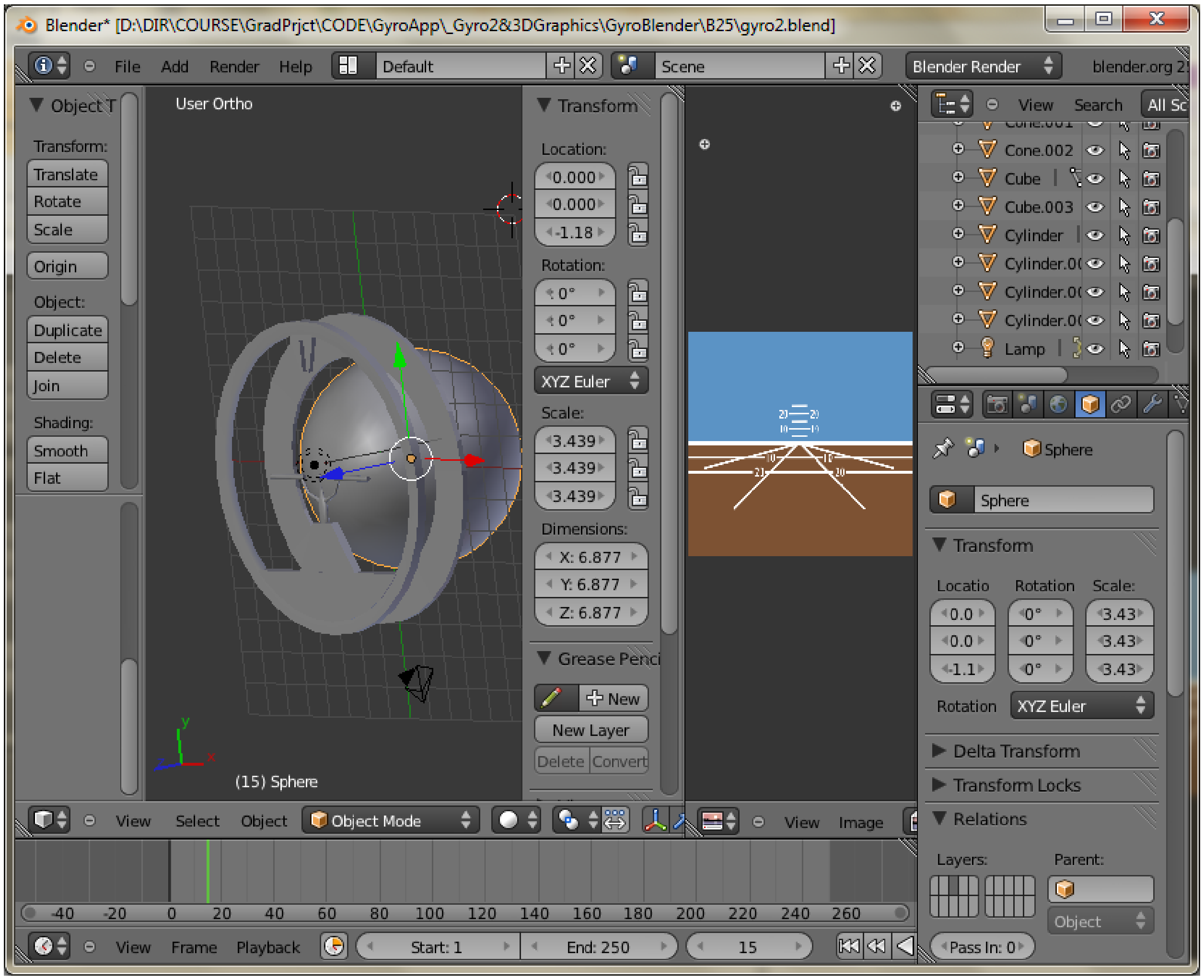


Fig. 55) Attitude Indicator 3D representation created in Blender v. 2.5 with 2D picture used to UV map the sphere part of the 3D object.

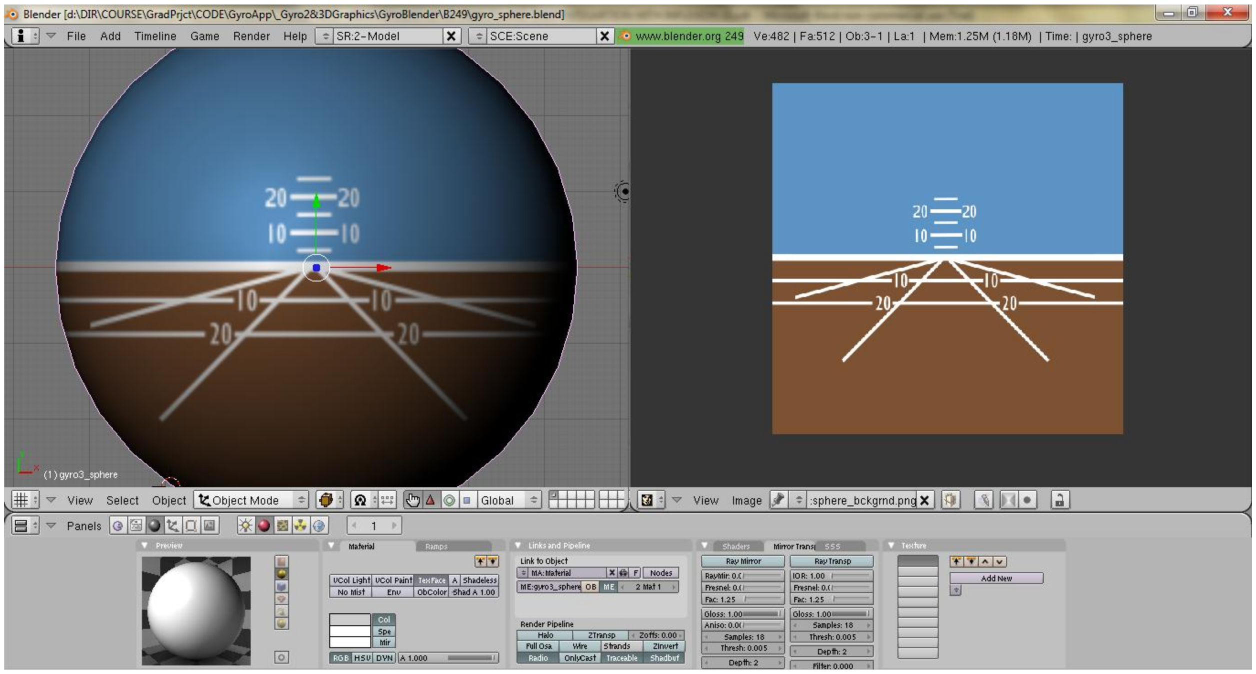

Blender [d:\DIR\COURSE\GradPrjct\CODE\GyroApp_Gyro2&3DGraphics\GyroBlender\B249\gyro_sphere.blend]
File Add Timeline Game Render Help SR:2-Model SCE:Scene www.blender.org 249 Ve:482 | Fa:512 | Ob:3-1 | La:1 | Mem:1.25M (1.18M) | Time: | gyro3_sphere
(1) gyro3_sphere
View Select Object Object Mode Global View Image sphere_bckgrnd.png
Panels

Fig. 56) Partial (sphere) mapping of the Attitude Indicator 3D object (imported by Blender v. 2.49)

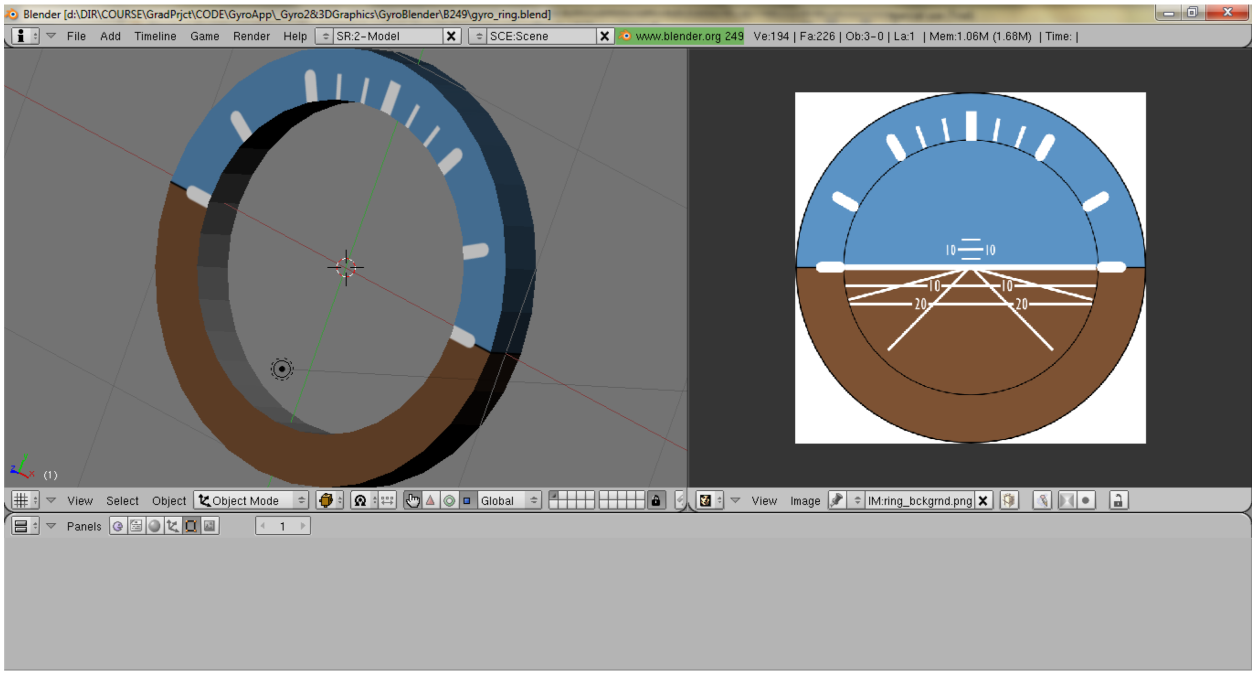

Blender [d:\DIR\COURSE\GradPrjct\CODE\GyroApp_Gyro2&3DGraphics\GyroBlender\B249\gyro_ring.blend]
File Add Timeline Game Render Help SR:2-Model SCE:Scene www.blender.org 249 Ve:194 | Fa:226 | Ob:3-0 | La:1 | Mem:1.06M (1.68M) | Time: |
View Select Object Object Mode Global View Image IM:ring_bckgrnd.png
Panels

Fig. 57) Partial (ring) mapping of the Attitude Indicator 3D object (imported by Blender v. 2.49)

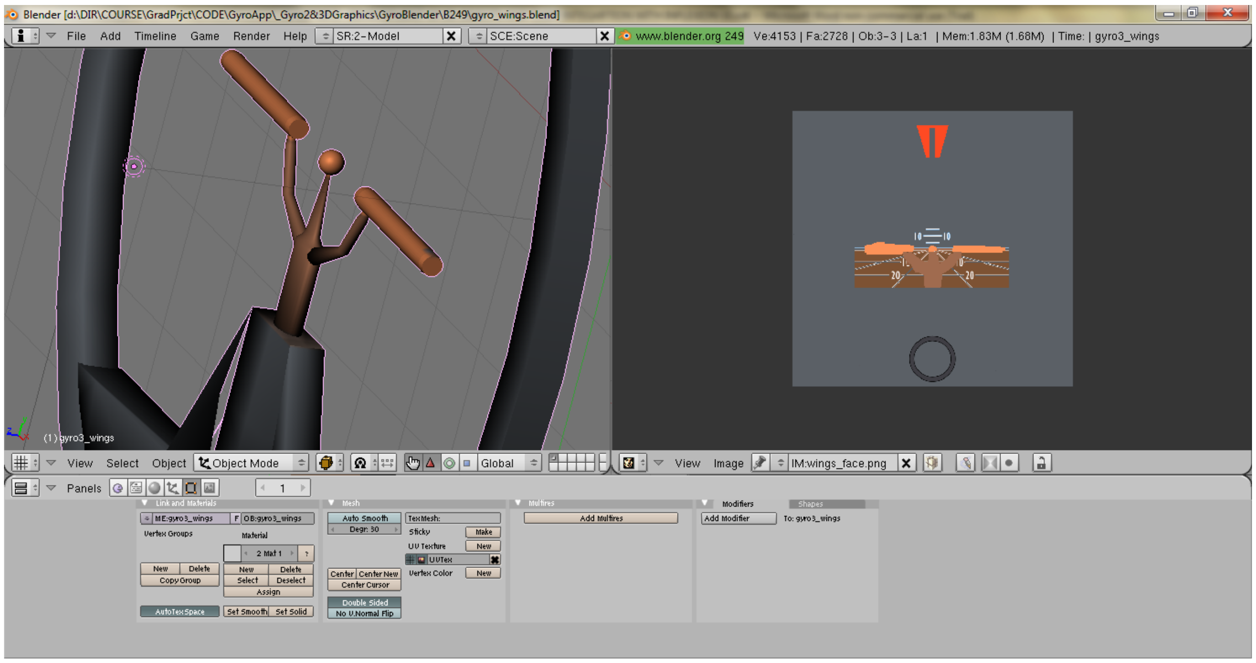
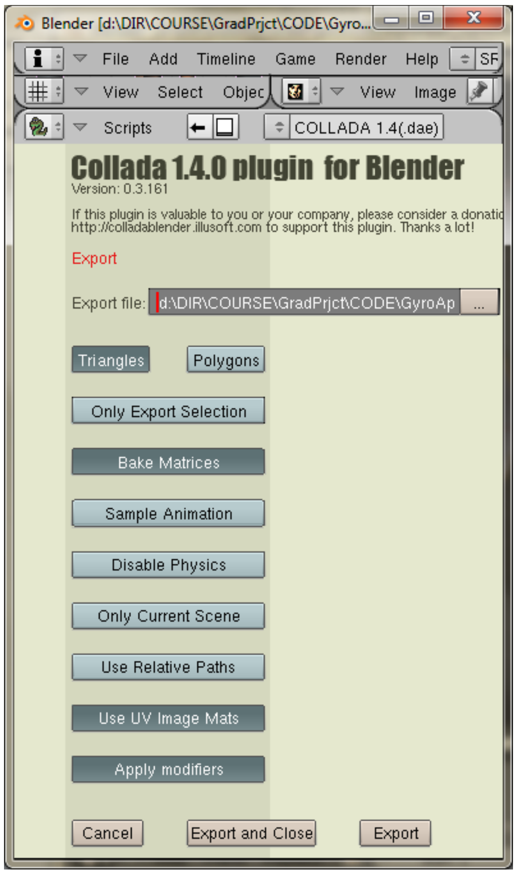

Blender [d:\DIR\COURSE\GradPrjct\CODE\GyroApp_Gyro2&3DGraphics\GyroBlender\B249\gyro_wings.blend]
File Add Timeline Game Render Help
SR:2-Model
SCE:Scene
www.blender.org 249 Ve:4153 | Fa:2728 | Ob:3-3 | La:1 | Mem:1.83M (1.68M) | Time: | gyro3_wings
(1) gyro3_wings
View Select Object Object Mode Global
View Image IM:wings_face.png
Panels

Fig. 58) Partial (wings) mapping of the Attitude Indicator 3D object (Blender v. 2.49 import is shown in Fig. 59)


Blender [d:\DIR\COURSE\GradPrjct\CODE\Gyro...
File Add Timeline Game Render Help
View Select Objec
View Image
Scripts
COLLADA 1.4(.dae)

Collada 1.4.0 plugin for Blender
Version: 0.3.161

If this plugin is valuable to you or your company, please consider a donatic
http://colladablender.illusoft.com to support this plugin. Thanks a lot!

Export

Export file: d:\DIR\COURSE\GradPrjct\CODE\GyroAp ...

Triangles | Polygons

Only Export Selection

Bake Matrices

Sample Animation

Disable Physics

Only Current Scene

Use Relative Paths

Use UV Image Mats

Apply modifiers

Cancel | Export and Close | Export

Fig. 59) Export of the 3D object from Blender v. 2.49 to Collada (.dae) file using settings acceptable by Inflexion UI (shown)

        - Rigging – a computer animation technique that groups elements in an animated 3D computer model. Object is represented by two elements: object's surface (mesh) and interconnected skeleton (rig).
        - Skinning - the process of creating the link between the rig and the mesh
        - Animation created in Blender has not been used for this project. Animation has been incorporated into the Android application solely using Inflexion UI Express.
        - Simulation – the process of imitating physical phenomena
        - Scripting – the way to extend Blender’s functionalities
        - Rendering – the process of converting object’s model into the image of its 2D representation
        - Compositing – combining separate elements into single object
        - Game creation – using integrated Blender’s gameengine to build interactive 3D applications
    - Uses OpenGL GUI that, uniform on all platforms and customizable through python scripts. Supported OS: XP, Vista, Win7, Linux, OS X, FreeBSD, Sun and others.
    - High quality 3D architecture enabling efficient work-flow
    - User community support by forums for questions, answers, and critique at http://BlenderArtists.org and news services at http://BlenderNation.com
    - Small executable size and easy distribution
- Blender’s Hot-Keys (used as the primary tool to access Blender’s functionalities). In-depth reference is available at http://download.blender.org/documentation/BlenderHotkeyReference.pdf

5.3. Using 3D objects within Inflexion UI environment

5.3.1. Import.

In order to make 3D objects created in Blender and exported to Collada (.dae) format available from within Inflexion UI Express environment, those files simply need to be copied into “<Project Directory> Graphics” folder using file manager (e.g. Windows Explorer). Accompanying images used for UV mapping earlier need to be copied into the same location. After copying, right click on "<Inflexion UI> GyroUI" shown in Fig. 60 and select "Refresh".

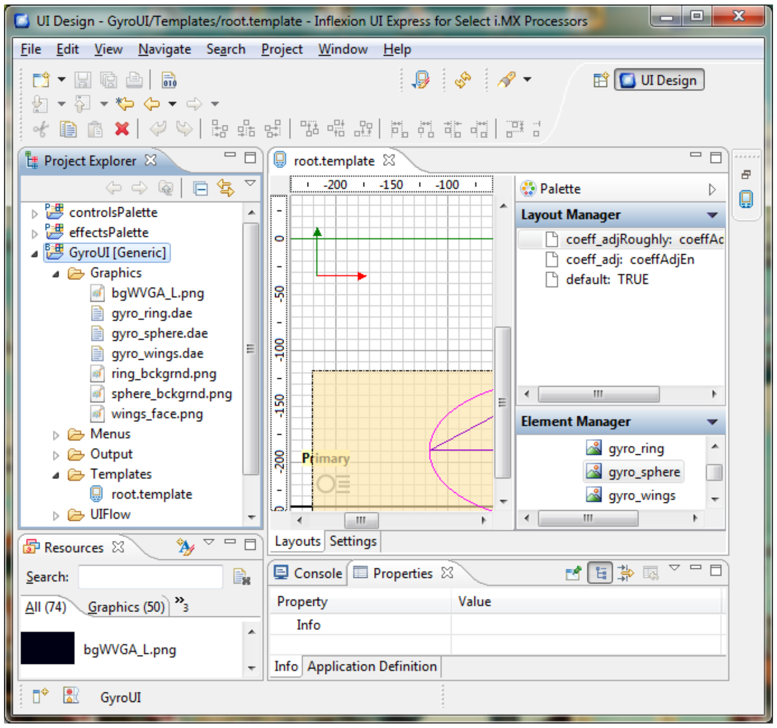
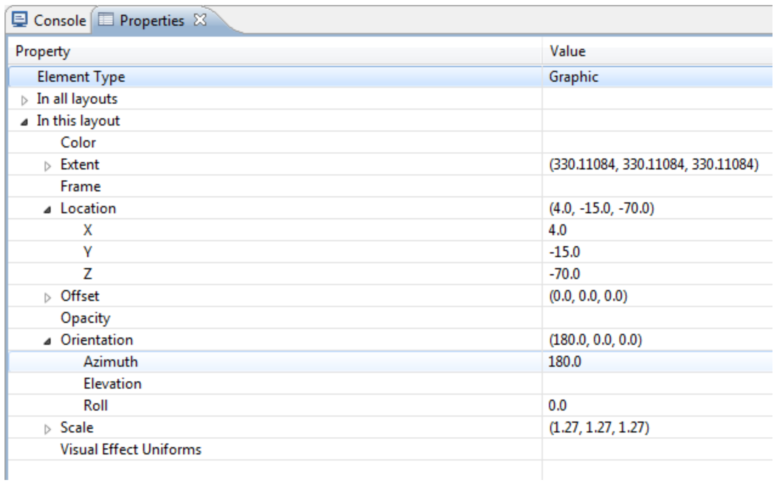


Fig. 60) 3D objects available from within Inflexion UI (.dae and .png files under "Project Explorer")

5.3.2. Once 3D objects become available in Inflexion UI, they can be dragged and dropped at the desired locations on the "Screen/Page" (select through "Element Manager") of the "root.template".

5.3.3. 3D object's Properties.

Fig. 61 illustrates the primary 3D object's properties include its original location (X, Y, Z) on the layout the object is placed on, and its spatial orientation (Azimuth, Elevation, Roll).


Console | Properties

Property	Value
Element Type	Graphic
In all layouts	
In this layout	
Color	
Extent	(330.11084, 330.11084, 330.11084)
Frame	
Location	(4.0, -15.0, -70.0)
X	4.0
Y	-15.0
Z	-70.0
Offset	(0.0, 0.0, 0.0)
Opacity	
Orientation	(180.0, 0.0, 0.0)
Azimuth	180.0
Elevation	
Roll	0.0
Scale	(1.27, 1.27, 1.27)
Visual Effect Uniforms	

Fig. 61) Primary 3D object properties available in Inflexion UI

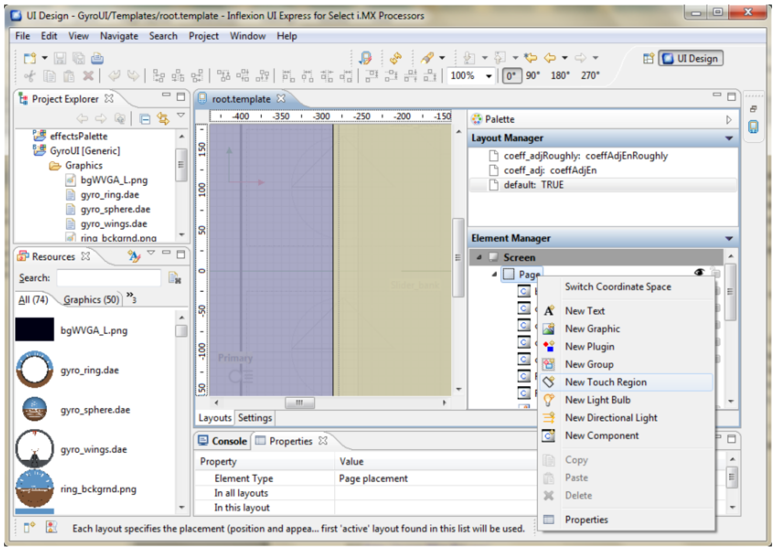
5.3.4. Animation.

An animation of the 3D object that constitutes the part of the user interface created in Inflexion IDE can be realized through its displacement. That way the object's original position (location and orientation, but also size, etc.), and hence appearance in the layout, can be interactively modified through updating assigned variable(s) values that control particular displacement. Those values are usually dynamically defined by Android application, outside the Inflexion UI, but also they can be set by Inflexion UI if variables are defined as local to that environment.

In order to setup 3D object's displacement and assign it to a given variable to following steps can be taken as an example:

- The variable to be used to displace a 3D object needs to be defined. It can be either Inflexion's UI local variable, not available from Android application developed in Eclipse IDE or a variable defined in Eclipse available for Inflexion UI. The process of defining variables has been described in Chapter 4 "Android Application" of this report.

- Using an Inflexion nomenclature, so called "Touch Region" needs to be created. The user gains a control over a 3D displacement through that "Touch Region". The "Touch Region" needs to be added as a component of the Page and placed, by dragging and dropping, on the active layout including that Page afterwards, as shown in Fig. 62.



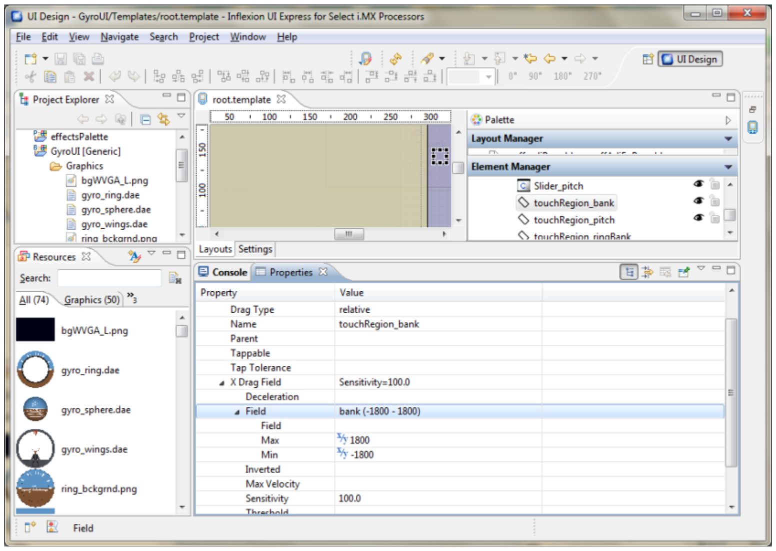
coeff_adjRoughly: coeffAdjEnRoughly
coeff_adj: coeffAdjEn
default: TRUE

Property	Value
Element Type	Page placement
In all layouts	
In this layout	

Each layout specifies the placement (position and appea... first 'active' layout found in this list will be used.

Fig. 62) Placement of a "Touch Region" component on the active layout using "Element Manager"

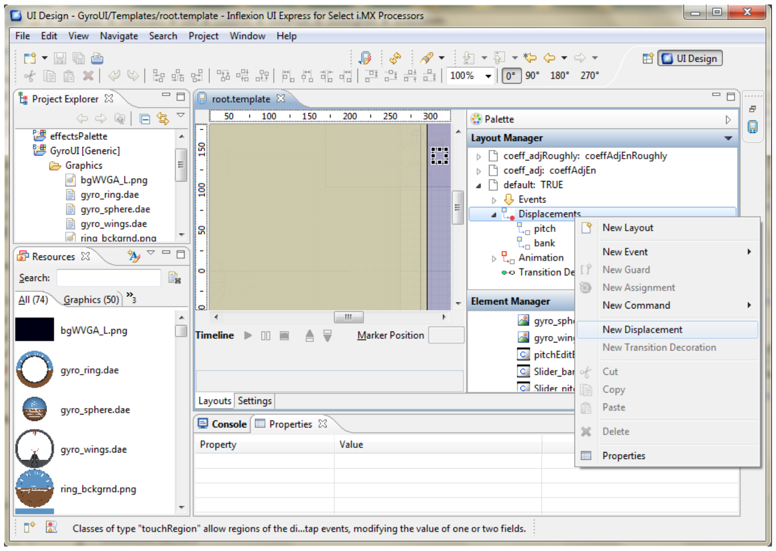
- The "Touch Region" has to be configured. Access to its properties is available through "Element Manager", as shown in the Fig. 63. The crucial properties of the "Touch Region" are the variable assigned to define its "Drag Field" and the operating range it would affect.



Property	Value
Drag Type	relative
Name	touchRegion_bank
Parent	
Tappable	
Tap Tolerance	
X Drag Field	Sensitivity=100.0
Deceleration	
Field	bank (-1800 - 1800)
Field	
Max	1800
Min	-1800
Inverted	
Max Velocity	
Sensitivity	100.0
Threshold	

Fig. 63) Access to "Touch Region" properties through "Element Manager"

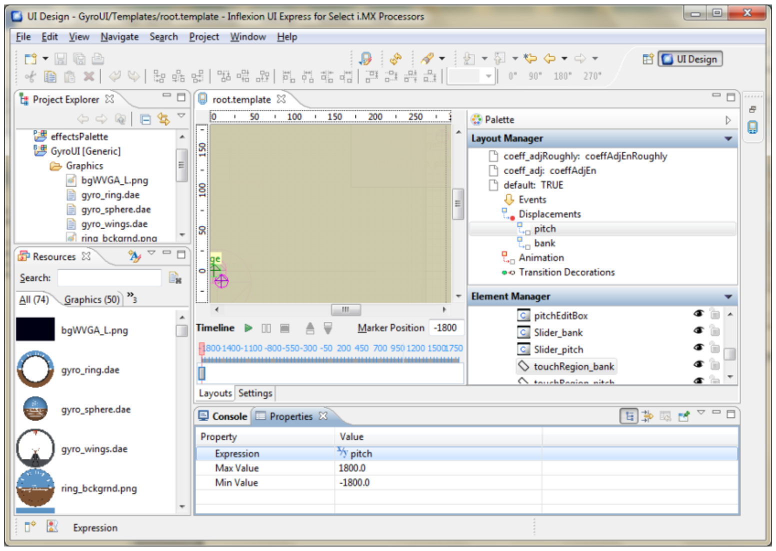
- After the variable to control displacement and the "Touch Region" are both set up, the "displacement" itself needs to be created and configured. The function of the "displacement" is to link the object's placement, which describes its appearance on the layout, with the variable that controls it dynamically (at runtime) within predefined range of movement. The displacement is created through “Layout Manager” as depicted in the Fig. 64:



Classes of type "touchRegion" allow regions of the di...tap events, modifying the value of one or two fields.

Fig. 64) Displacement setup access through “Layout Manager”

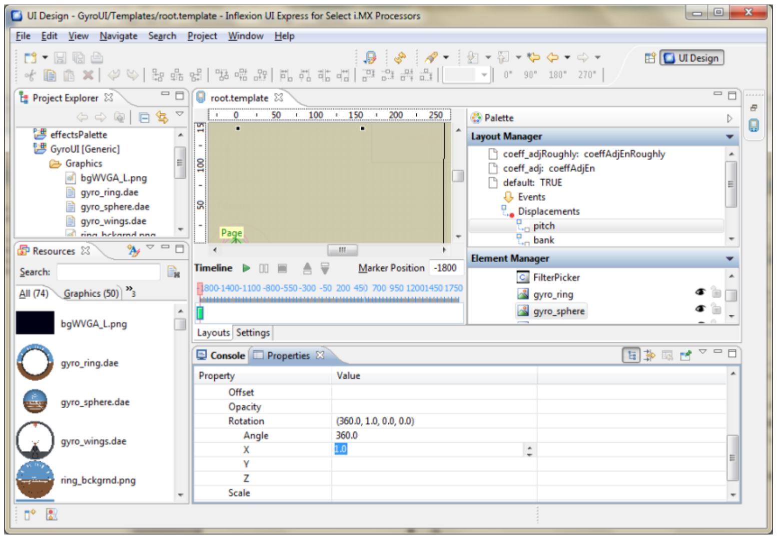
In order to assign the variable to drive the particular “displacement“, it needs to be entered as a displacement’s property along with corresponding range. Fig. 65 illustrates those settings.


UI Design - GyroUI/Templates/root.template - Inflexion UI Express for Select i.MX Processors
Layout Manager
Displacements
pitch
bank
Element Manager
Timeline
Marker Position -1800

Property	Value
Expression	pitch
Max Value	1800.0
Min Value	-1800.0

Fig. 65) The assignment of the variable “pitch” to drive a displacement in the range of -1800 ÷ 1800, i.e. 360deg in both directions with 0.1 degree resolution for smooth transitions within a real range of -180 ÷ 180 degrees.

- The displacement needs to be finally configured to drive the particular object, as depicted in Fig. 66. To do so, the displacement needs to be single clicked on, followed by single clicking on the object to create the link with. This enables an access to a different properties set then when object is accessed separately. The animation can be simulated using the “Timeline” feature at the bottom of the layout editor.


UI Design - GyroUI/Templates/root.template - Inflexion UI Express for Select i.MX Processors

Layout Manager
- coeff_adjRoughly: coeffAdjEnRoughly
- coeff_adj: coeffAdjEn
- default: TRUE
- Events
- Displacements
  - pitch
  - bank

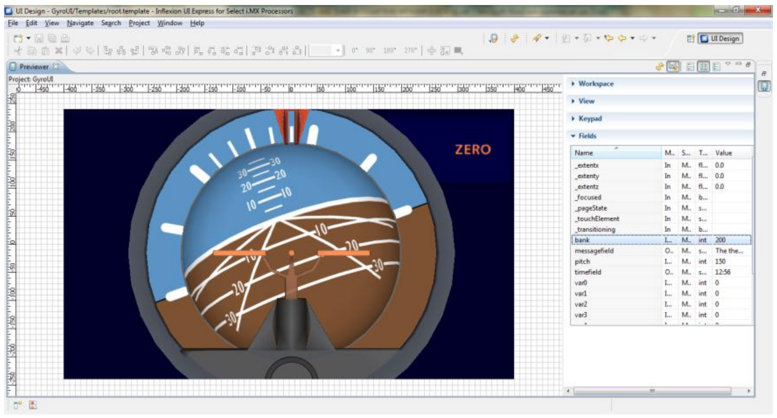
Element Manager
- FilterPicker
- gyro_ring
- gyro_sphere

Timeline — Marker Position -1800

Property	Value
Offset	
Opacity	
Rotation	(360.0, 1.0, 0.0, 0.0)
Angle	360.0
X	1.0
Y	
Z	
Scale	

Fig. 66) Linking “pitch” displacement with “gyro_sphere” 3D object and defining 360 degrees rotation of the object around its X axis to be controlled by the variable previously assigned to that displecement.

- Finally the object can be seen in action using previewer which provides a developer to the direct access to all the variables, those local and those set by the Android application at runtime. Fig. 67 illustrates the previewer. Previewer allows the test of the interactive user interface before installing it on the physical Android device.


UI Design - GyroUI/Templates/root.template - Inflexion UI Express for Select i.MX Processors

Fields

Name	M..	S...	T...	Value
_extentx	In	M..	fl...	0.0
_extenty	In	M..	fl...	0.0
_extentz	In	M..	fl...	0.0
_focused	In	M..	b...	
_pageState	In	M..	s...	
_touchElement	In	M..	s...	
_transitioning	In	M..	b...	
bank	I...	M..	int	200
messagefield	O..	M..	s...	The the...
pitch	I...	M..	int	150
timefield	O..	M..	s...	12:56
var0	I...	M..	int	0
var1	I...	M..	int	0
var2	I...	M..	int	0
var3	I...	M..	int	0

Fig.67) Inflexion Previewer

Chapter 6

USB Interface

6.1. Overview

6.1.1. USB (Universal Serial Bus) is a popular standard interface used by USB devices (e.g. Android tablets) to communicate with a USB host (e.g. PC).

With the introduction of microcontrollers incorporating the USB OTG (On-The-Go) module by Microchip, it became possible for embedded applications to utilize the wide range of USB devices as a USB embedded hosts, using their chips.

Per USB standard specification, USB devices cannot communicate directly with each other as they need to communicate with USB host that controls the USB bus through which one or more devices exchange data. [34]

The USB host has to learn about the USB device and assign a device driver to handle further communication over a USB bus. USB device enumeration process is defined by the following steps[37] :

- Device is plugged into host’s USB port
- USB hub detects the device
- Host gets notified about new device attached to the bus through an interrupt
- Hub determines if the device is a low or high speed
- Hub resets the device
- Host learns if full speed device supports high speed
- Hub establishes the signal path between USB device and the bus
- Hosts sends a request packet to learn the maximum packet size of the default pipe. It uses Get_Descriptor request for this purpose.
- Host assign an address to the device
- Host learns about device’s features
- Host assigns a device driver
- Host loads a device driver
- The driver selects device’s configuration and its interface(s) are enabled ever since for communication

6.1.2. USB interface optional implementations.

- Tethering Android device to USB device, i.e.: i.MX53 as a USB host and uController as a USB device interoperated via USB bus.

  Majority of Android devices acts as USB devices that must be connected to and controlled by USB host (e.g. PC) via star USB bus. Through tethering, an Android device becomes a USB host that controls USB devices on the bus. Such a host can connect to USB devices, in the contrary to the USB device that cannot connect to other USB devices. Fig. 68 depicts USB Host-Device topology.

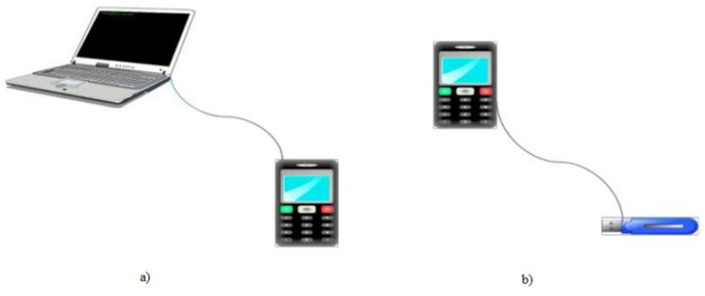


Fig. 68) Android device as a USB device (a). Android device as a USB host (b). [35]

In order to transform a standard USB device running on Android OS, its drivers need to be modified to support USB OTG interface. In case such modification is made by an independent developer, it would be lost after uploading the new revision of the operating system to the Android device by its vendor, who does not maintain that driver as a part of device's OS kernel. Therefore custom USB OTG solution involves the following challenges [35] :

- USB host has to supply 5VDC to USB devices per its specification. Standard Android device is not equipped in the hardware to support this requirement, so customization is necessary.

- In Android environment, USB services are supported by device drivers, as depicted in Fig. 69, and Linux kernel. USB host uses number of drivers that logically communicate with a given device driver that handles external, physical USB equipment. Each Android device is built out of a device-specific hardware, with unique registers, data buffers, intra-hardware dependencies, etc. The hardware of each, vendor/PN/revision dependent chipset, differs and hence requires specific host controller driver to handle each particular device, with its unique features and without affecting the remaining, standardized part of the USB stack, i.e. host core driver that incorporates common, across host core drivers, solutions to handle standard USB functions like buffer management, devices' attachment and configuration, data transfers, etc.

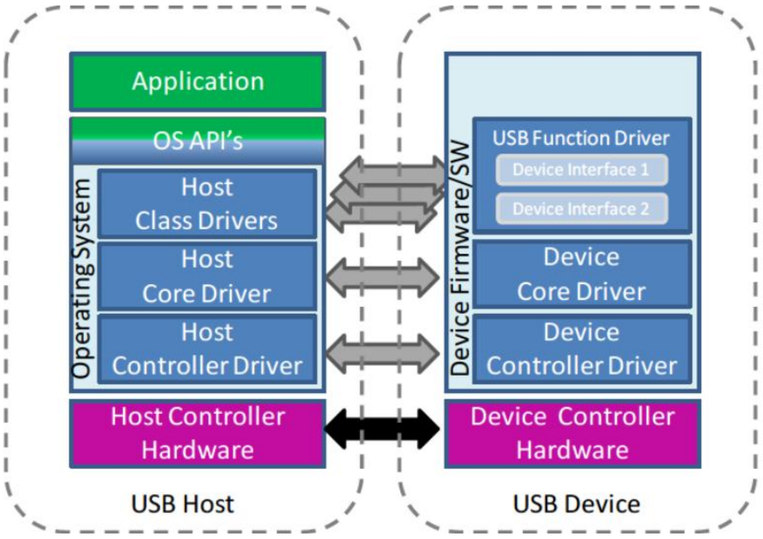


Fig. 69) USB driver architecture [35]

Function drivers encapsulate device interfaces. USB host needs to access USB device interface(s) to obtain information about device's capabilities. USB core driver and host controller, both handle devices enumeration, connection procedures and data exchange. Enumerated USB device reports to the host its configuration(s) that describe(s) device's interface(s) and endpoint(s) to connect to. USB function drivers serve the USB host as access solutions to device functionalities. USB host has host class drivers implemented, which it uses to communicate with USB device function drivers. Linux OS supports standardized, e.g. HID, CDC, etc. class drivers for the number of interfaces. New capabilities that come with USB OTG specification require new driver architecture that needs to support host and device mode during and after having a USB communication channel established. In order for the Android device to provide USB host capabilities, the processor running OS has to have USB host controller hardware built-in or at least available through available on-board interfaces. Also, the processor needs a driver for Linux in order to act as a USB host. BSP (Board Support Package) with Android OS used for this project, provided by Adeneo includes USB host controller drivers, but access to those drivers is not available from the Android API level, and therefore such implementation would involve additional Android OS level modifications and kernel recompilation. These changes would negatively impact future generations of the solution, if any, as Adeneo would possibly provide new (higher) API levels of Android OS designed for i.MX53 platform that would not include those custom modifications. Thus, every OS update would involve further kernel modifications and recompilations. For those reasons this approach was dropped.

- "Open Accessory API or Open Accessory Framework - this is the API/framework in the Android development environment that allows the Android applications to transmit data in and out of the available USB port.

This is provided by Google through the Android SDK.” [36] The Open Accessory solution seemed to be the simplest and the most direct with full support in the Android development environment, but at the time of developing this project, Adeneo was providing BSP with Android 2.2 which did not include Open Accessory framework. For that reason the approach was dropped.

- ADB (Android Debug Bridge) interface, i.e.: i.MX53 as a USB device interoperated via USB bus and uController as a USB host. ADB has been implemented on every Android device since its early introduction. It defines de-facto standard for debugging Android devices. It was found that all features of the ADB would not be needed for the purpose of implementing communication channel between i.MX53 and external board with uController and MPU-6000 populated on it. ADB functionality of port forwarding via TCP (Transmission Control Protocol) channel has been implemented on i.MX53 Android device configured as a USB device and ADB server and on uController configured as a USB host and ADB client. Existing USB debugging socket on the i.MX53 board has been used for communication purposes over ADB channel.

6.1.3. USB Hosts and Peripheral Devices.

Per USB specification, USB host is responsible for supplying 5VDC to the USB device(s) on the other end over the USB cable to announce that it is connected to that(those) device(s). Linux OS constitutes the core of Android OS, and hence services provided by Android devices use Linux services and kernel. Android device’s hardware uses Linux kernel as well as its drivers and libraries. Therefore Android JAVA USB applications actually access Linux drivers to implement communication via USB channel.

A typical USB system consists of one host and one or more peripheral devices. Per common nomenclature they are referred to as USB devices. Particular USB device can communicate directly with the host that acts as a centre of USB tiered star network topology. Thus, USB devices have no way to establish and maintain communication channel with each other.

USB devices have a mechanism implemented to send data to the host only after the host requests it. The host indicates to the device when it’s ready to accept data and at the same time the device needs to be capable to accept data incoming from the host.That way all communication on the bus is always initialized by the host controlling the traffic.

USB devices are usually divided into categories. Those categories are called classes within USB terminology. Classes have special requirements as for their communication format so that the USB host can recognize them. The host needs to meet the requirements of the given class in order to establish USB communication session. Device drivers provide API to handle classes on the application level. Classes might be standard (e.g. HID-Human Interface Device like mouse or keyboard) or vendor specific that require special (non-standard) drivers that require separate USB client drivers.

"The number of devices that can attach to a host can be expanded through the use of hubs. Typically, a hub allows four or seven devices to attach to a single port. A maximum of five hubs can be chained together, creating up to five tiers. A maximum of 127 devices (including the hubs) can be connected on the bus. A full USB host uses a Type-A receptacle, and must be able to communicate with any device. This support may be provided via special drivers that must be installed on the host prior to attaching the device. Hubs must be supported, and each port must be able to deliver a minimum of 100 mA." [34]

6.1.4. Host Mode

USB devices respond to requests initiated by the USB host, which controls all the traffic on the bus. USB devices are not capable of initiating data transfers.

The USB OTG (On The Go) module, that PIC24FJ256GB106 is equipped with, has been used in the host mode adequate for this project. In general, it could be configured as a USB device as well per OTG specification.

USB transfers consist of, usually, multiple transactions, that on the other hand consist of multiple packets. Control transfers in most cases require all transactions. Interrupt, isochronous and bulk transfers do not use neither "SETUP" token nor the status transaction. Bulk transfers allow for the transference of up to 64 bytes within single data stage transaction. [34] Fig. 70 illustrates a USB transfer state machine.

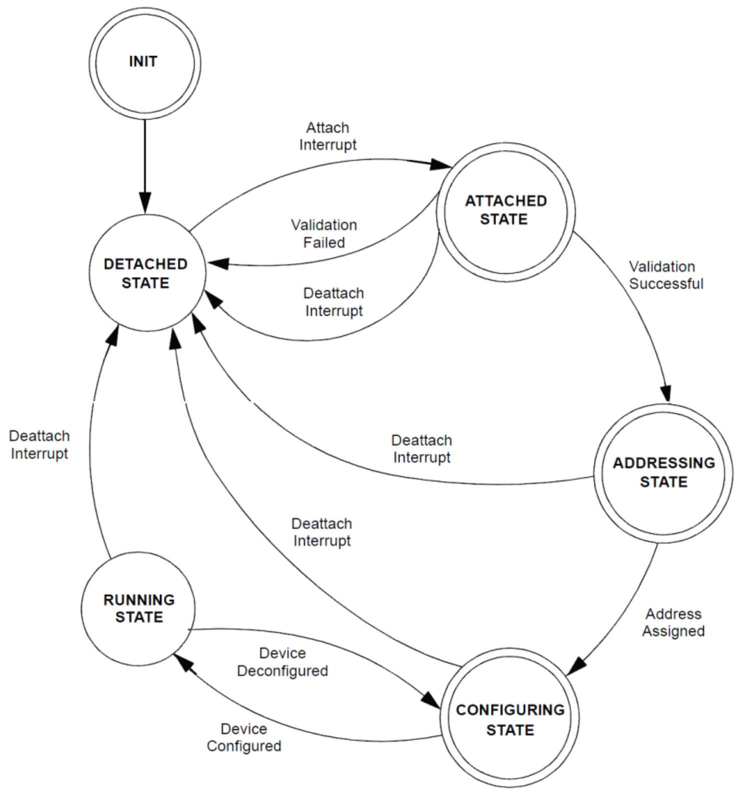


Fig. 70) USB embedded host state machine - Format of a single USB transfer [34]

6.2. ADB (Android Debug Bridge).

ADB is a debugging protocol implemented on all Android OS since Android's introduction. ADB defines the rules that control data exchange between ADB client (e.g. PC) and ADB server running on actual Android device via USB interface. Through ADB, the Android device can provide ADB client with a shell access, and hence direct execution of the command(s) defined in ADB specification available at [7]. The feature of TCP ports forwarding via sockets enables the establishment of bidirectional pipes between an Android device and ADB client. In that case, an Android application listens on the port acting as an ADB server, while the PIC24 connects to that port as an ADB client. In this case embedded application corresponds to the PC. ADB provides the set of communication channels that enable the host to open a session with Android device physically connected to it via USB cable. Android devices use ADB communication channels to access services ADB provides, such as:

- Data forwarding over ADB channel to a TCP socket, also called "port forwarding".

  The service enables the process running on Android OS to use TCP sockets API, so it can listen and accept connections on the particular TCP port. This feature of the ADB protocol has been used to implement USB communication between microcontroller by Microchip on board, serving as a mid-man between MPU-6000 chip and an Android device, and i.MX53 acting as an Android device itself.

Port forwarding service forwards communication data supposed to target localhost on the specified port to the external Android device with ADB server running on it via USB interface. It works the similar way in an opposite direction, while port numbers do not need to match. ADB bridge may be seen as a link between client TCP socket on the host (e.g. PC) side and server TCP socket on the Android device side. Server socket implemented on PIC24FJ256GB106 listens for incoming connection establishment requests.

Attitude Indicated developed for this project uses communication interfaces as depicted in Fig. 71.

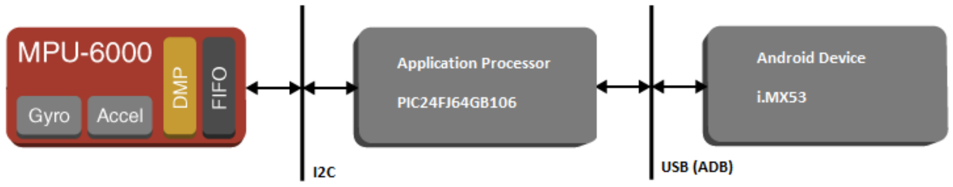
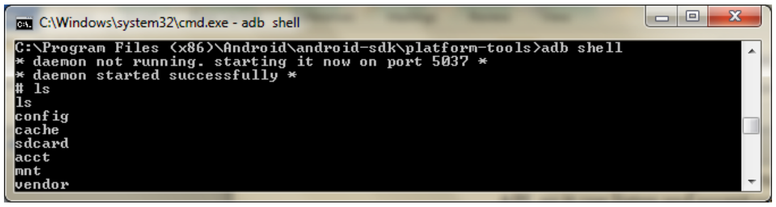


Fig. 71) Chain of interfaces utilized in the Aircraft Attitude Indicator application

- Linux shell
- File system access (shown in Fig. 72)



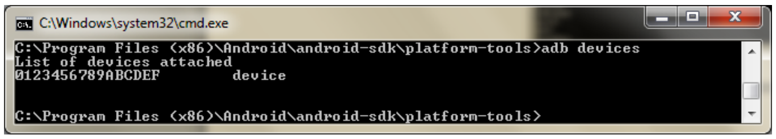
```
C:\Program Files (x86)\Android\android-sdk\platform-tools>adb shell
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
# ls
ls
config
cache
sdcard
acct
mnt
vendor
```

Fig. 72) Linux shell and file system access on Android device through Windows “cmd” and ADB service

- Debugging functions

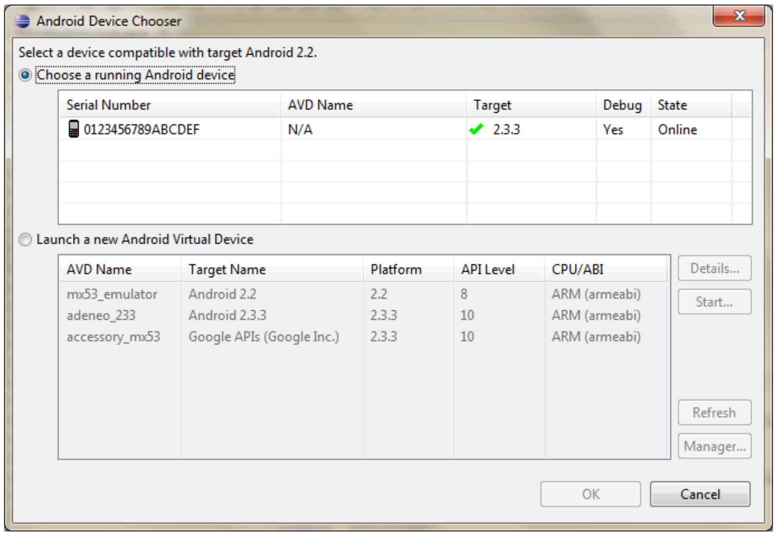
In order to receive the full list of functions/options "adb" command should be executed without any arguments. Some of the ADB functions are:

- ▪ Listing connected devices (shown in Fig. 73-74)



```
C:\Program Files (x86)\Android\android-sdk\platform-tools>adb devices
List of devices attached
0123456789ABCDEF        device

C:\Program Files (x86)\Android\android-sdk\platform-tools>
```

Fig. 73) ADB launch through Windows’ “cmd”


Android Device Chooser

Select a device compatible with target Android 2.2.

(●) Choose a running Android device

Serial Number	AVD Name	Target	Debug	State
0123456789ABCDEF	N/A	✓ 2.3.3	Yes	Online

( ) Launch a new Android Virtual Device

AVD Name	Target Name	Platform	API Level	CPU/ABI
mx53_emulator	Android 2.2	2.2	8	ARM (armeabi)
adeneo_233	Android 2.3.3	2.3.3	10	ARM (armeabi)
accessory_mx53	Google APIs (Google Inc.)	2.3.3	10	ARM (armeabi)

Fig. 74) Access to Android device serial number via ADB protocol through Windows "cmd" and through Eclipse

- Connecting/disconnecting to/from a device via TCP/IP (if no port specified, 5555 is assumed)
- File operations, including root access
- Running remote shell
- Viewing device's log (corresponds to Eclipse's logcat)
- Forwarding socket connections
- Pushing package files into the device
- Removing app packages from the device
- Returning all information from the device
- Starting/Killing of the ADB server

6.3. Debugging feature of the Android device.

In order to use ADB, a debugging needs to be enabled on the Android device, usually through running "Settings>Applications>Development>USB debugging" on that device.

6.4. Main components of the Android implementation of the USB device ADB server.

The implementation of the ADB interface for Android application is straightforward and relies on regular TCP communication implementation similar to the following in brief (part of PitchBankModula.java):

```
private class usbCommHandler implements Runnable {
        public void run() {
                ...
```

```
server_ = new ServerSocket(PORT_NO);
...
while (true) {
        ...
        socket_ = server_.accept();
        ...
        while ((socket_.getInputStream().read(input_data)) != -1) {
        ...socket_.getOutputStream().write(output_data);
        ...
```

6.5. Embedded implementation of the USB host / ADB client

The firmware for this project has been written for the PIC24 series processor by Microchip. It constitutes a modification of the IOIO (pronounced yo-yo) project available at [39] . ADB (Android Debug Bridge) specification itself is available at [7]. Information regarding the implementation of the ADB on the embedded platform available at [38] was found very helpful along with the resources accessible from [39] and Microchip’s USB library that had a direct impact on the final shape of the communication-via-USB-interface related part of the firmware.

Microchip's library provides USB host stack, i.e. an implementation of the USB host driver that includes handler for USB embedded host hardware interface and application interfaces for client drivers. The library defines constants, data types, structures and macros, common to the multiple layers of the Microchip USB Firmware Stack and the USB Device Framework protocol described in a Chapter 9 of the USB 2.0 specification. IOIO [7] provides Android driver for a USB embedded host device as well as support for ADB protocol. For the purpose of this project, the driver files had been modified to support PIC24FJ254GB106 processor. Moreover the driver defines common ADB layer types and implements an ADB packet transfer mechanism and API on top of the USB layer that enables data exchange between ADB server and client.

On the top level, the ADB protocol is handled by the state machine. The main part of the handler is as follows (the part of “main.c” file):

```
. . .
switch(state) {
    case MAIN_STATE_WAIT_CONNECT:
     if (connected) {
       print0("ADB connected!");
       h = ADBOpen("tcp:4356", &ChannelRecv);
       state = MAIN_STATE_WAIT_READY;
     }
     break;

    case MAIN_STATE_WAIT_READY:
     if (ADBChannelReady(h)) {
      state = MAIN_STATE_RUN;
    }
    break;

   case MAIN_STATE_RUN:
        if (mpu6000_data.pitch != mpu6000_data.pitch_mem ||
        mpu6000_data.bank != mpu6000_data.bank_mem ||
        delta(&ADBReconnect_dwInternalTicks_mem, FALSE) > 1.0) {
      ADBWrite(h, &mpu6000_data, 4);
      mpu6000_data.pitch_mem = mpu6000_data.pitch;
      mpu6000_data.bank_mem = mpu6000_data.bank;
      state = MAIN_STATE_WAIT_READY;
    }
    break;

    default: break;
  }
}
. . .
```

ADBOpen() – opens a channel for the remote ADB server, with the TCP port defined during the call. ChannelRecv() serves as a receive handler (handles data incoming from i.MX53). In case the connection request is refused by the remote end, the ADBChannelReady() returns appropriate value to indicate it through the API as it does in case the request to establish a communication channel via ADB results in a "ready" state. The role of ADBChannelReady() function is to indicate the success or the failure in opening the channel between ADB server and client, and to notify the API whether the channel is ready for a data transmission. ADBWrite() writes data to the remote end (i.MX53) via open channel and indicates whether the data have been received on that end or not through the acknowledge mechanism.

Chapter 7

Conclusions

The project demonstrates the implementation of the Aircraft Attitude Indicator that uses USB ports to interface Android application with the cutting edge MPU-6000 gyroscope / accelerometer from Invensense. An Android based implementation of the Aircraft Attitude Indicator has been presented with the source code and all design details including hardware schematic. An extensive research was done in order to combine application specific requirements, including 3D graphics and its animation, with the outputs of the processor. Those outputs were calculated during motion processing of the readings from the gyroscope and the accelerometer. Android Debug Bridge protocol was studied to establish a communication link between an external board designed for this project with gyroscope/accelerometer populated on it and an i.MX53 development board with Android OS loaded onto.

Android platform has been found suitable for an aircraft instrument application as it provides wired USB interface, virtually limited merely by designer's imagination graphics development environment in conjunction with Inflexion IDE from Mentor Graphics and might possibly be equipped with a large touch screen.

It has been found vital for the success of the Attitude Indicator implementation to develop reliable Motion-Fusion algorithm that combines the readings of the gyroscope and accelerometer. A digital filter design was identified as the most difficult part of that phase of the project. To reliably combine gyroscope's and accelerometer's readings with simultaneous removal of the linear acceleration in order to obtain pure gravity for further motion processing, turned out to be crucial for the development of the device that could be proven to work in the field. It was assumed that Invensense delivered the solution that provides a near "perfect" estimate of the spatial position of the aircraft, i.e. Motion-Fusion library that in part runs in encrypted form on the MPU (Motion Processing Unit). The main problem encountered throughout the completion of the project was the Invensense IP in the form of the library and its migration to the target PIC24 platform. At some point Invensense removed the library from their website. It can only be assumed that complains from the developers forum might have triggered it to happen. Consequently, due to problems with Invensense's IP, the approach had to be changed and some concrete steps taken in order to finalize the project in the timely manner. As the result, motion-fusion algorithms were formulated and entered into the PIC24 processor firmware that handled processed data exchange between MPU-6000 and i.MX53. The final solution chosen for this project is independent from Invensense and can interface with a different gyro / accel device(s) with few modifications as it handles MPU-6000 now, but does not utilize Invensense library. As the future consideration, it is worth to note that modifying and integrating the library for Atmel processor that Invensense currently makes available on their website, might turn out beneficial for the Attitude Indicator performance as Invensense is perceived to possess an extensive experience in the field and therefore might be able to provide the best solutions available at the moment.

Working on the project let practice JAVA and C/C++ programming for the prolonged time and familiarize with most up to date development tools, which has been found extremely beneficial for the future work in an engineering field. All the functional modules such as: Android application with 3D graphics linked to particular variables, processor <–> i.MX53 data exchange via USB interface, processor <–> MPU-6000 data exchange via I^2C interface and a motion processing including filtration of the signals were simulated and tested individually and during the final phase of the design combined together.

Overall, considerable amount of knowledge was gained through the process of developing the solution that involved integration of few separate development environments such as in the area of 3D objects design using popular Blender, their animation and integration with Android application in Eclipse through Inflexion UI. The latter allows skipping Open GL techniques and detailed knowledge necessary for efficient development of 3D user interface for Android application and hence was found a very helpful tool to speed up a time to market. Familiarity with Invensense cutting edge solutions, MPU-60X0 in particular has been acquired as well as detailed knowledge about motion processing.

References

[1] "Airplane Attitude Instrument Flying" Chapter 4, Section II "Using an Electronic Flight Display"
[2] http://en.wikipedia.org/wiki/Flight_instruments; March 2012
[3] “FTF-ENT-F0541.pdf” document by freescale
[4] "PIC24FJ256GB110 Family Data Sheet" by Microchip
[5] “MPU-6000/MPU-6050 EV Board User Guide” by Invensense
[6] http://www.invensense.com/; March 2012
[7] http://www.freescale.com/; March 2012
[8] http://www.mentor.com/; March 2012
[9] http://www.adeneo-embedded.com/; March 2012
[10] http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en531078; March 2012
[11] PS-MPU-6000A.pdf available @ [6]
[12] http://www.mathworks.com/help/toolbox/aeroblks/f3-22568.html; March 2012
[13] “Aircraft Motion Analysis” J.A.Thelander; March 2012
[14] http://www.flightlearnings.com/2010/09/30/attitude-indicator/; March 2012
[15] http://overtheairwaves.com/vol3-20final.html; March 2012
[16] “MPU-6000 and MPU-6050 Product Specification Revision 3.2” by Invensense
[17] http://www.adeneo-embedded.com/; March 2012
[18] http://invensense.com/mems/about.html; March 2012
[19] “Atmel UC3 Reference Implementation User Guide for InvenSense Embedded MotionApps™ Platform Release 2.0.0” by Invensense
[20] http://invensense.com/mems/faq.html; March 2012
[21] http://en.wikipedia.org/wiki/Coriolis_effect; March 2012
[22] "DEVELOPMENT OF HIGH-PERFORMANCE, HIGH-VOLUME CONSUMER MEMS GYROSCOPES" by Joe Seeger, Martin Lim, and Steve Nasiri
[23] “Android Development for Embedded Systems Beyond Mobile” by Colin Walls, Mentor Graphics
[24] “Lab: Deploy the application on the device” by freescale & Adeneo Embedded
[25] “Android SDK and tools” by freescale and Adeneo Embedded
[26] “Android application files” by freescale and Adeneo Embedded
[27] http://www.arm.com/files/pdf/11mentor_android_gui_challenges.pdf; March 2012
[28] "How to Quickly Invigorate Your Device Utilizing 3D UI Technology" by Mentor Graphics
[29] http://www.blender.org/; March 2012
[30] http://www.w3schools.com/xml/xml_usedfor.asp; March 2012
[31] http://wiki.blender.org/index.php/Doc:2.6/Manual/Introduction; March 2012
[32] http://en.wikipedia.org/wiki/UV_mapping; November 2011
[33] http://en.wikipedia.org/wiki/Polygon_mesh; November 2011
[34] “USB Embedded Host Stack” by Microchip www.microchip.com
[35] “Tethering an Android Smartphone to USB Devices” www.securecommconsulting.com; November 2011
[36] “Microchip's Accessory Framework for Android(tm)” www.microchip.com; November 2011

[37] "USB Complete Fourth Edition - The Developers Guide" Jan Axelson
[38] http://code.google.com/p/microbridge/; October 2011
[39] https://github.com/ytai/ioio/wiki; November 2011
[40] http://developer.android.com/guide/developing/tools/adb.html; March 2012
[41] http://developer.android.com/resources/dashboard/platform-versions.html; March 2012
[42] "Instrument Flying Handbook" by Federal Aviation Administration, Skyhorse Publishing

Appendix A

Development Software Installation Guide

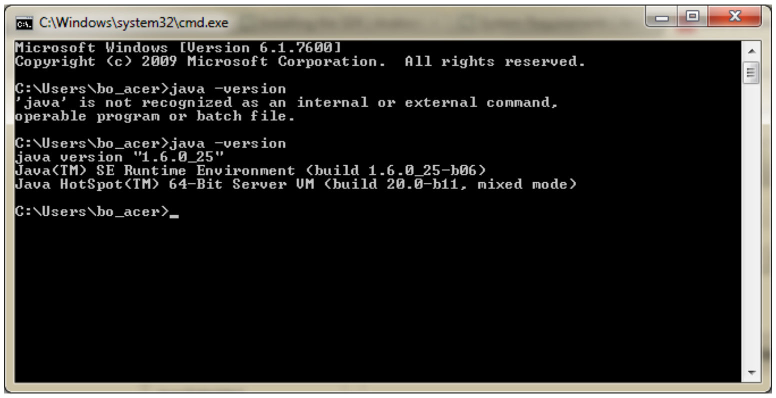
The following installation guide applies to Win7(x64), and describes, step by step, how the software components required to setup Eclipse IDE Android Development Environment and embedded C compiler should be integrated with the system.

Links and webpage screenshots used in this guide were updated in February, 2012.

During the entire installation process AVD Manager, SDK Manager, Inflexion UI and Eclipse IDE must be launched with Administrator privilages (in Win7, right click and select "Run as Administrator" instead of simply double clicking on the program's icon).
Internet connectivity should be available during installation process to allow automatic download of any necessary dependencies by particular software.

Development Computer preparation:

1. Download Java Development Kit (JDK) from "http://www.oracle.com/technetwork/java/javase/downloads/index.html" and install it on the host, if not yet installed (check installation status by launching "cmd" in Windows environment and executing a command: "java -version", as shown in Fig. 75.



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\bo_acer>java -version
'java' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\bo_acer>java -version
java version "1.6.0_25"
Java(TM) SE Runtime Environment (build 1.6.0_25-b06)
Java HotSpot(TM) 64-Bit Server VM (build 20.0-b11, mixed mode)

C:\Users\bo_acer>_
```

Fig. 75) Java version check

2. Let Installation Wizard for JAVA SE Development Kit (JDK) guide through the installation process. Leave all the options as default.

3. Download Eclipse IDE (Interactive Development Environment) from "http://www.eclipse.org/downloads/" and extract it to the desired destination folder. Eclipse version supported by current Android SDK is listed at "http://developer.android.com/sdk/requirements.html". At the time of writing this guide Eclipse 3.6 (Helios) and greater are being supported. Google recommends to install one of the packages for developing Android applications. Following this recommendation is needed to properly integrate ADT (Android Development Tools) into suitable Eclipse IDE. Per

"http://developer.android.com/sdk/eclipse-adt.html": "The Eclipse Classic version is recommended. Otherwise, a Java or RCP version of Eclipse is recommended.":

- Eclipse Classic – a version used for this project. Eclipse Classic Package includes the Eclipse Platform, Java Development Tools, and Plug-in Development Environment with source and both user and programmer documentation.

- Eclipse IDE for Java Developers

- Eclipse IDE for Java EE Developers

4. Eclipse JDT plugin must be either included in Eclipse IDE package (it is included in Eclipse Classic 3.6.2 used to develop Android application for this project) or it must be installed separately. The JDT project provides the tool plug-ins that implement Java IDE supporting the development of any Java application, including Eclipse plug-ins. It adds a Java project nature and Java perspective to the Eclipse Workbench as well as a number of views, wizards, editors and builders. Furthermore, it includes merging and refactoring tools. The JDT project allows Eclipse to become a development environment for itself.

5. Launch Eclipse IDE and pick the location of the workspace as desired.

6. Download Android SDK from "http://developer.android.com/sdk/index.html" and intall it on the host. During installation process use default options.

7. Start SDK manager to install most up to date packages via internet. Use default selections. (Add Android 2.2 platform as it is used to develop 3D app that runs on i.MX53 which uses Android OS provided by Adeneo through BSP (Board Support Package). BSP related information is provided in next point of this guide as well.

8. Install the ADT Plugin for Eclipse IDE (without this step Eclipse could not be used as a development environment for Android applications). Per "http://developer.android.com/sdk/eclipse-adt.html": Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to give a powerful, integrated environment to build Android applications. ADT extends the capabilities of Eclipse to allow to set up new Android projects quickly, to create an application UI (User Interface), add components based on the Android Framework API (Application Programming Interface), debug applications using Android SDK tools, and export signed/unsigned .apk files in order to distribute given application. ADT provides guided project setup, tools integration, custom XML editors, debug ouput pane, and basically gives an incredible boost in developing Android applications. Before installing or using ADT, a compatible version of Eclipse and at least one development platform must be installed on a development computer, as explained above. Revisions of Eclipse IDE as well as Android SDK must both match those enumerated for the current ADT (16.0.1) at "http://developer.android.com/sdk/eclipse-adt.html" URL location, i.e. Eclipse Helios (Version 3.6) or higher and Android SDK Tools r16 (at the time of writning this guide).

   Use Eclipse Update Manager feature to install ADT:

- Start Eclipse, then select "Help > Install New Software"...
- Click "Add", in the top-right corner.

- In the "Add Repository" dialog that appears, enter "ADT Plugin" for the "Name" and the following URL for the "Location": "https://dl-ssl.google.com/android/eclipse/"

- Click "OK". Note: If you have trouble acquiring the plugin, try using "http" in the "Location" URL, instead of "https".

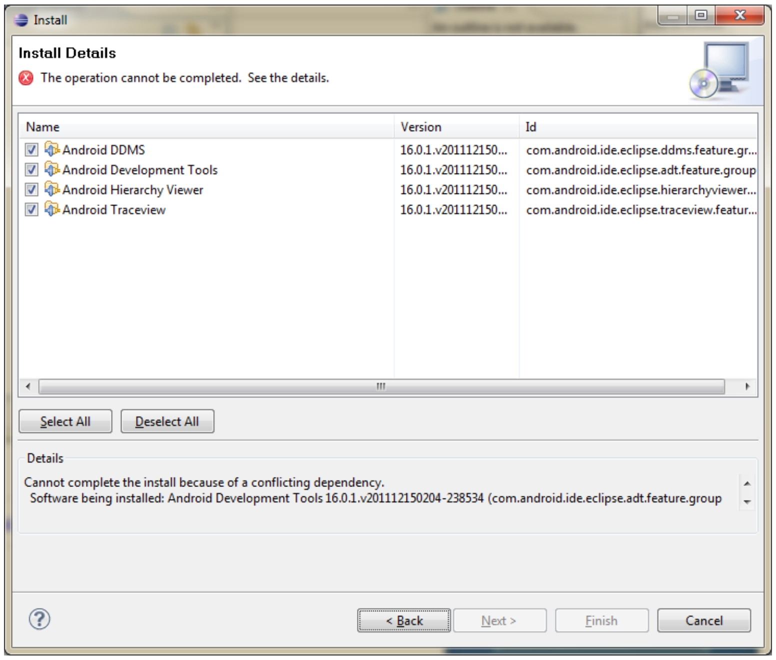
- In the "Available Software" dialog, select the checkbox next to "Developer Tools" and click "Next".

- In the next window, a list of the tools to be downloaded will be enumerated. Click "Next".

- In order to prevent appearance of the message as shown in Fig. 76, the entries must be added through "Eclipse/Help/Install New Software/Available Software Sites/..."

"Name: Helios
Location: http://download.eclipse.org/releases/helios"

"Name: The Eclipse Project Updates
Location: http://download.eclipse.org/eclipse/updates/3.6"



Install Details

The operation cannot be completed. See the details.

Name	Version	Id
✓ Android DDMS	16.0.1.v201112150...	com.android.ide.eclipse.ddms.feature.gr...
✓ Android Development Tools	16.0.1.v201112150...	com.android.ide.eclipse.adt.feature.group
✓ Android Hierarchy Viewer	16.0.1.v201112150...	com.android.ide.eclipse.hierarchyviewer...
✓ Android Traceview	16.0.1.v201112150...	com.android.ide.eclipse.traceview.featur...

Details

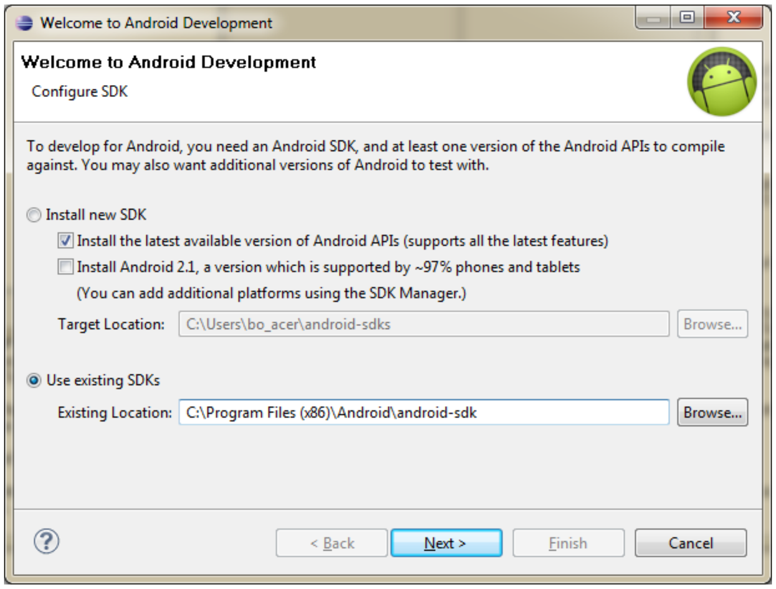
```
Cannot complete the install because of a conflicting dependency.
 Software being installed: Android Development Tools 16.0.1.v201112150204-238534 (com.android.ide.eclipse.adt.feature.group
```

Fig. 76) ADT plugin install details

- Click "Finish". Note: If you get a security warning saying that the authenticity or validity of the software can not be established, simply click "OK".

- When the installation completes, restart Eclipse IDE.

- After restarting Eclipse IDE, select "Use existing SDKs" and provide the path to SDK installation on the host in order to modify ADT preferences. Follow an example as shown in Fig. 77. Update the path to SDK installation if required.



Welcome to Android Development

Configure SDK

To develop for Android, you need an Android SDK, and at least one version of the Android APIs to compile against. You may also want additional versions of Android to test with.

( ) Install new SDK
[✓] Install the latest available version of Android APIs (supports all the latest features)
[ ] Install Android 2.1, a version which is supported by ~97% phones and tablets
(You can add additional platforms using the SDK Manager.)
Target Location: C:\Users\bo_acer\android-sdks

(●) Use existing SDKs
Existing Location: C:\Program Files (x86)\Android\android-sdk

Fig. 77) ADT preferences

- Optionally ADT preferences can be changed at any time by following the procedure:

  - Select "Window > Preferences..." to open the "Preferences" panel

- Select Android from the left panel. Click "Proceed" after making the choice regarding sending usage statistics to Google.

- Click "Browse..." and locate directory containing SDK.

- Click "Apply" followed by clicking "OK".

9. Add platforms and other components.

- In order to modify SDK setup use the Android SDK (Software Development Kit) and AVD (Android Virtual Device) Manager (a tool included in the SDK starter package) to download essential SDK components into your development environment.

  The SDK uses a modular structure that separates the major parts of the SDK

Android platform versions, add-ons, samples, tools, and documentation into a set of components installed separately. To develop an Android application, at least one Android platform and the associated platform tools need to be installed. Other components and platforms can be added as well. For this project Android application uses Android Platform 2.2, API level 8 (one of the most popular Android platforms in January-February 2012). Fig. 78 illustrates Android's market popularity.

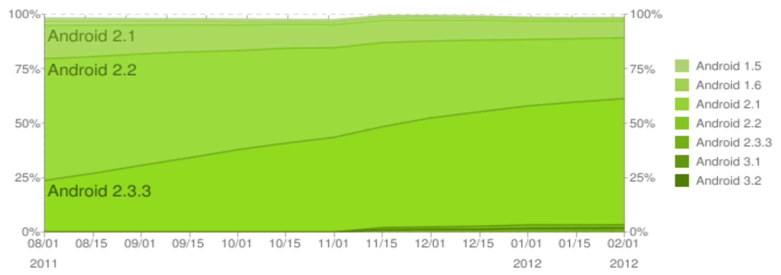
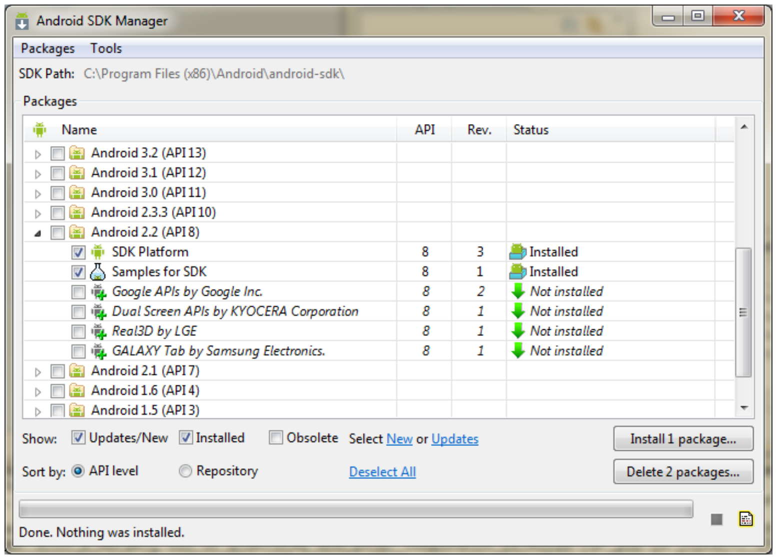


Fig. 78) [41] Last historical dataset collected during a 14-day period ending on February 1, 2012 (based on the number of Android devices that have accessed Android Market)

- Launch the Android SDK, shown in Fig. 79, and AVD Manager in one of the following ways:

From within Eclipse, select "Window > Android SDK Manager / AVD Manager." SDK and AVD Managers has been separated in the newest revision of the SDK. AVD Manager setup is not necessary in case a target Android device is availale (for this project i.MX53 was available to begin with and therefore a virtual device has not been used). Debugging on bith virtual and real devices is possible with the use of the same tools and approach so no particular benefit has been noticed coming from using the virtual one. Add required platforms as follows:



SDK Path: C:\Program Files (x86)\Android\android-sdk\

Packages

Name	API	Rev.	Status
Android 3.2 (API 13)			
Android 3.1 (API 12)			
Android 3.0 (API 11)			
Android 2.3.3 (API 10)			
Android 2.2 (API 8)			
✓ SDK Platform	8	3	Installed
✓ Samples for SDK	8	1	Installed
Google APIs by Google Inc.	*8*	*2*	*Not installed*
Dual Screen APIs by KYOCERA Corporation	*8*	*1*	*Not installed*
Real3D by LGE	*8*	*1*	*Not installed*
GALAXY Tab by Samsung Electronics.	*8*	*1*	*Not installed*
Android 2.1 (API 7)			
Android 1.6 (API 4)			
Android 1.5 (API 3)			

Show: [✓ Updates/New] [✓ Installed] Obsolete Select New or Updates

Sort by: [✓ API level] Repository

Done. Nothing was installed.

Fig. 79) Android SDK Manager

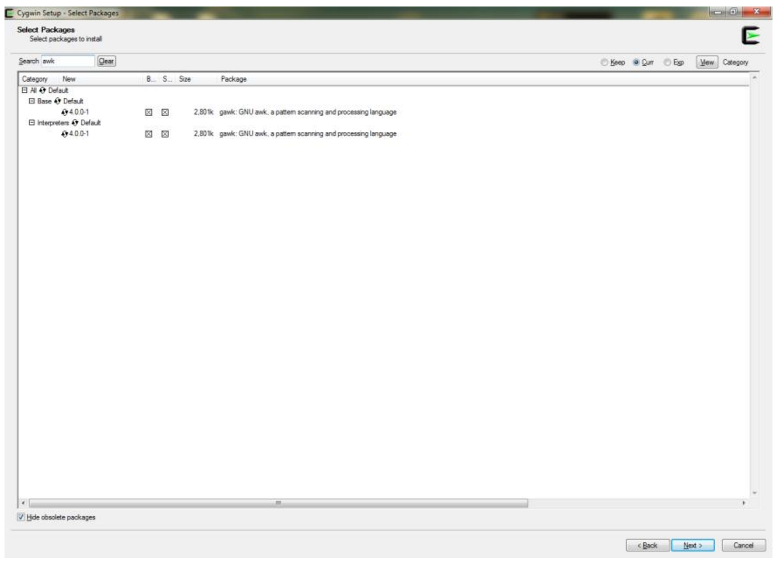
10. Installation of Inflexion UI (User Interface) by Mentor Graphics.

- Inflexion makes use of native API application programming interface (written in C/C++) in Android application(s), so that development of the application takes place directly on the target platform without involvement of the virtual machine, and hence the Android NDK (Native Development Kit) which is a toolset that allows generating libraries from C/C++ sources and embedding them into an application package file (.apk) that can be deployed on Android device needs to be integrated with Android SDK.

- Install Cygwin 1.7 or higher in Windows. Cygwin is a Unix-like environment and command-line interface for Microsoft Windows. Cygwin provides native integration of Windows-based applications, data, and other system resources with applications, software tools, and data of the Unix-like environment.

  - Download Cygwin from "http://www.cygwin.com/" and launch "setup.exe". Use default options during installation, except:

    - select "awk" version 3.1.8 or newer under packages during Cygwin pre-installation setup as shown in Fig. 80.



Cygwin Setup - Select Packages

Select Packages
Select packages to install

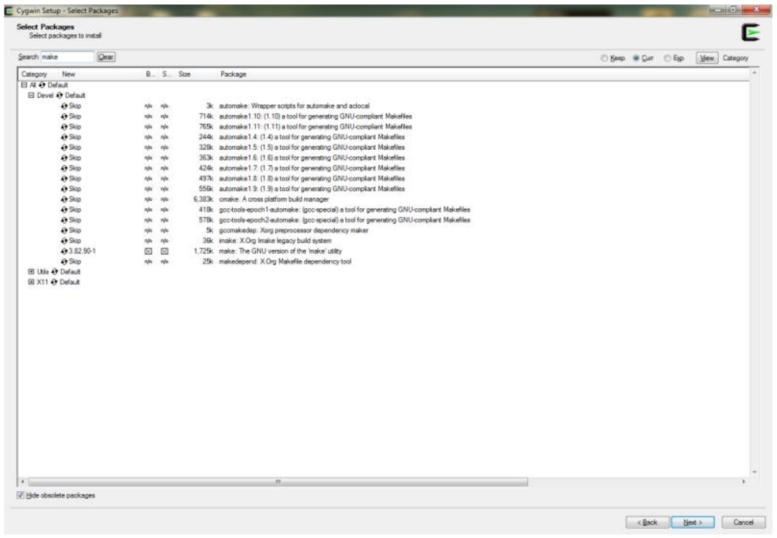
Search: awk
Keep [✓ Curr] Exp
View: Category

Category	New	B...	S...	Size	Package
⊟ All ⟳ Default					
⊟ Base ⟳ Default					
	⟳ 4.0.0-1	☒	☒	2,801k	gawk: GNU awk, a pattern scanning and processing language
⊟ Interpreters ⟳ Default					
	⟳ 4.0.0-1	☒	☒	2,801k	gawk: GNU awk, a pattern scanning and processing language

[✓] Hide obsolete packages

Fig. 80) Cygwin: awk setup

    - select "make" version 3.81 or newer under packages during Cygwin pre-installation setup as shown in Fig. 81.


Cygwin Setup - Select Packages
Select Packages
Select packages to install
Search make Clear
Keep [✓ Curr] Exp View Category
Hide obsolete packages
< Back Next > Cancel

Category	New	B...	S...	Size	Package
⊟ All Default					
⊟ Devel Default					
	Skip	n/a	n/a	3k	automake: Wrapper scripts for automake and aclocal
	Skip	n/a	n/a	714k	automake1.10: (1.10) a tool for generating GNU-compliant Makefiles
	Skip	n/a	n/a	765k	automake1.11: (1.11) a tool for generating GNU-compliant Makefiles
	Skip	n/a	n/a	244k	automake1.4: (1.4) a tool for generating GNU-compliant Makefiles
	Skip	n/a	n/a	328k	automake1.5: (1.5) a tool for generating GNU-compliant Makefiles
	Skip	n/a	n/a	363k	automake1.6: (1.6) a tool for generating GNU-compliant Makefiles
	Skip	n/a	n/a	424k	automake1.7: (1.7) a tool for generating GNU-compliant Makefiles
	Skip	n/a	n/a	497k	automake1.8: (1.8) a tool for generating GNU-compliant Makefiles
	Skip	n/a	n/a	556k	automake1.9: (1.9) a tool for generating GNU-compliant Makefiles
	Skip	n/a	n/a	6,383k	cmake: A cross platform build manager
	Skip	n/a	n/a	418k	gcc-tools-epoch1-automake: (gcc-special) a tool for generating GNU-compliant Makefiles
	Skip	n/a	n/a	578k	gcc-tools-epoch2-automake: (gcc-special) a tool for generating GNU-compliant Makefiles
	Skip	n/a	n/a	5k	gccmakedep: Xorg preprocessor dependency maker
	Skip	n/a	n/a	36k	imake: X.Org Imake legacy build system
	3.82.90-1	☒	☒	1,725k	make: The GNU version of the 'make' utility
	Skip	n/a	n/a	25k	makedepend: X.Org Makefile dependency tool
⊞ Utils Default					
⊞ X11 Default					

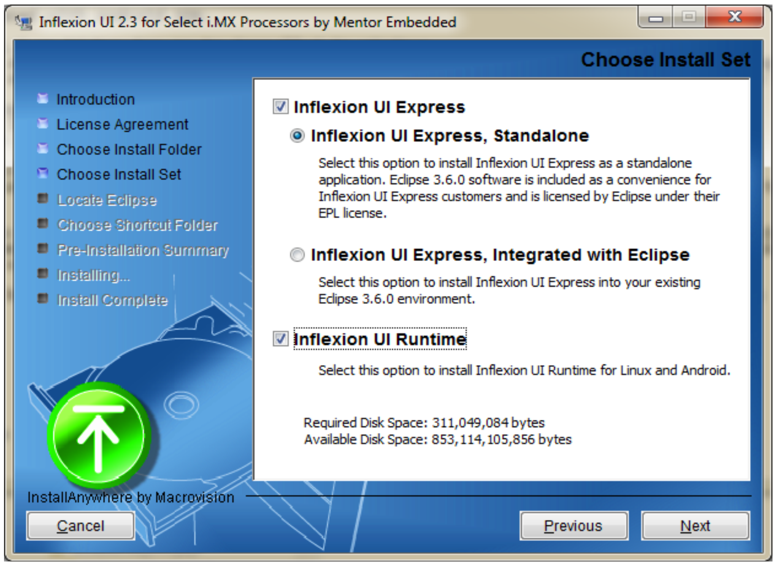
Fig. 81) Cygwin: make setup

Note: Neither "awk" nor "make" is included by default.

- Download NDK from "http://developer.android.com/sdk/ndk/index.html".

Extract the file to desired location. No further installation is needed. Tools will be updated with a path to NDK directory later.

- Download Inflexion UI from "http://www.mentor.com/embedded-software/inflexion/ui-imx-processors". For this project, i.MX53 Quick Start Reference Board by Freescale + TFT LCD display with touchscreen (optionally VGA graphics could be used without touchscreen capability) have been used as a target Android device.
- Install Inflexion UI (Rev.2.3 has been used for this project). Follow installer's default recommendations across installation process. Use settings as depicted in Fig. 82:



☑ Inflexion UI Express

[✓ Inflexion UI Express, Standalone]

Select this option to install Inflexion UI Express as a standalone application. Eclipse 3.6.0 software is included as a convenience for Inflexion UI Express customers and is licensed by Eclipse under their EPL license.

Inflexion UI Express, Integrated with Eclipse

Select this option to install Inflexion UI Express into your existing Eclipse 3.6.0 environment.

☑ Inflexion UI Runtime

Select this option to install Inflexion UI Runtime for Linux and Android.

Required Disk Space: 311,049,084 bytes
Available Disk Space: 853,114,105,856 bytes

Fig. 82) Inflexion UI: Express/Runtime installer launch

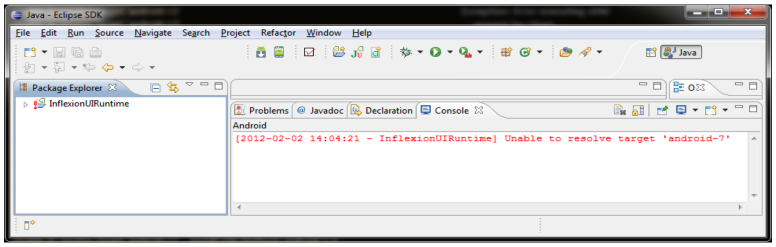
11. Configuration of Inflexion UI Android Environment.

- Install Inflexion Android Plugin into the Eclipse IDE.

  - Create "apps" folder under NDK installation directory.
  - Copy "inflexionui" folder from "<Inflexion UI install directory>/InflexionUIRuntime-2.3/Android/" to "<NDK install directory>/apps". InflexionUIRuntime-2.3 directory might have different name depending on the current revision of Inflexion UI that has been installed previously.
  - Launch Eclipse IDE and select "Help/Install New Software...". Click "Add...".
  - Type "Inflexion UI Project" in the "Name" field
  - Type "http://s3.mentor.com/inflexionplugin/freescale_2.2" in the "Location" field, and click "OK".
  - Click on "?" in bottom left corner and then click on "Select All" after having selected "Inflexion UI Project - http://s3.mentor.com/inflexionplugin/freescale_2.2" in "Work with" combo box, in case "Next" button is not enabled (grayed).
  - Click "Next" until "Finish" gets enabled, then click "Finish" button and follow defaults until installation finishes. Restart Eclipse IDE after installation process is over.
  - In Eclipse IDE, select "Window/Reset Perspective" and click "OK" when confirmation dialog box appears.
  - Select "Window/Pereferences/Inflexion UI"
  - Enter the root of NDK that contains "inflexionui" folder previously copied into it, e.g. "C:\Program Files (x86)\Android\android-ndk-r7".
  - Enter cygwin location, e.g. "C:\cygwin"
  - Click "OK".

12. Install Inflexion Runtime Library (Inflexion engine) on a target device (i.MX53). In order to install Android Application that includes User Interface/3D graphics integrated in Inflexion UI on the target Android device, Inflexion Engine (Runtime Library) needs first to be installed on that device. Runtime Library installed on the device makes this device capable to run applications created using Inflexion UI (package commonly called Inflexion PC tool). Any Inflexion UI based Android Application loads Inflexion UI Runtime Library at startup.

- In Eclipse IDE, select File/Import.
- Navigate to and select "General/Existing Projects into Workspace". Click "Next".
- Select root directory by browsing to and selecting "<NDK install directory>/apps/inflexionui/framework". Click "OK".
- Make sure "InflexionUIRuntime" is selected in the "Projects list" and click "Finish".
- In case the error message appears as shown in Fig. 83,



```
Android
[2012-02-02 14:04:21 - InflexionUIRuntime] Unable to resolve target 'android-7'
```

Fig. 83) Inflexion UI Runtime: Target error message

which depends on the current revision of Inflexion Runtime Library and Android API level installed in SDK, install missing API level as follows:

- Activate "Window/Android SDK Manager".
- Pick a package to install per the following guide:
  - Unable to resolve target 'android-1' ==> (Android 1.0) change the "AndroidManifest.xml"
  - Unable to resolve target 'android-2' ==> (Android 1.1) change the "AndroidManifest.xml"
  - Unable to resolve target 'android-3' - install SDK Platform Android 1.5
  - Unable to resolve target 'android-4' - install SDK Platform Android 1.6
  - Unable to resolve target 'android-5' - install SDK Platform Android 2.0
  - Unable to resolve target 'android-6' - install SDK Platform Android 2.0.1
  - Unable to resolve target 'android-7' - install SDK Platform Android 2.1
  - Unable to resolve target 'android-8' - install SDK Platform Android 2.2
  - Unable to resolve target 'android-9' - install SDK Platform Android 2.3
  - Unable to resolve target 'android-10' - install SDK Platform Android 2.3.3
  - Unable to resolve target 'android-11' - install SDK Platform Android 3.0
  - Unable to resolve target 'android-12' - install SDK Platform Android 3.1
  - Unable to resolve target 'android-13' - install SDK Platform Android 3.2
  - Unable to resolve target 'android-14' - install SDK Platform Android 4.0
  - Unable to resolve target 'android-15' - install SDK Platform Android 4.0.3

- Click "Install...".

- Try again.
  - In Eclipse IDE, select "File/Import"
  - Navigate to and select "General/Existing Projects into Workspace". Click "Next".
  - Select root directory by browsing to and selecting "<NDK install directory>/apps/inflexionui/framework". Click "OK".
  - Right click on "InflexionUIRuntime" in "Eclipse/Package Explorer" and Fix Project Properties as shown in Fig. 84:

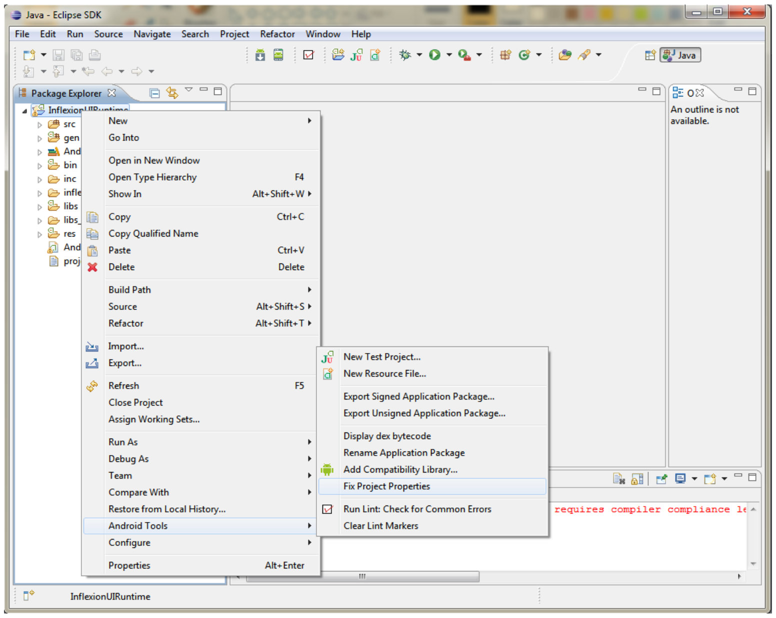
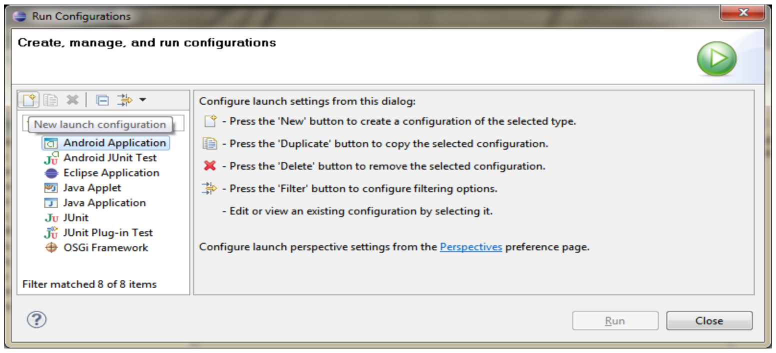


Fig. 84) Project properties automatic fix

- In "Eclipse/Package Explorer", right click on the "InflexionUIRuntime" and select "Run As/Run Configurations...".
- Select "Android Application" as shown in Fig. 85 and click on "New launch" configuration icon (one of the icons shown right above filter text box):


Run Configurations

Create, manage, and run configurations

Configure launch settings from this dialog:
- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the Perspectives preference page.

Fig. 85) Run configuration

- On the "Android" tab:
  - Enter "RunInflexionUIRuntime" into the "Name" field.
  - Click "Browse" button and pick "InflexionUIRuntime".
- On the Target Tab:
  - Set "Deployment Target Selection Mode" to "Manual".
  - Click "Apply".

        - Click "Run".

- Select one the the devices compatible with target Android platform by selecting "Choose a running Android device" and then selecting the one actually connected via USB interface. Normally it should be just one position on the list, unless more external devices use ADB channel via separate USB interfaces. For i.MX53 the device Serial Number is: 0123456789ABCDEF.

- Provided i.MX53 is physically connected via USB interface with the host PC, the library will be loaded onto the target device (loading the library onto the device need to happen only one time per device), otherwise the message will appear informing the user that no compatible targets were found. Click "OK".

- Target Android Device (i.MX53) is now ready to launch Android Applications with user interface/graphics developed in Inflexion PC tool and integrated with Android application code (JAVA, C/C++) in Eclipse IDE. Now, a developer needs to launch Inflexion UI, develop desired user interface using imported (custom) 3D object(s), previously developed in Blender and exported into Collada format, and/or objects available through Inflexion UI library. Then he needs to integrate Inflexion output files with Android application developed in Eclipse and load it onto Target device using Eclipse IDE features.

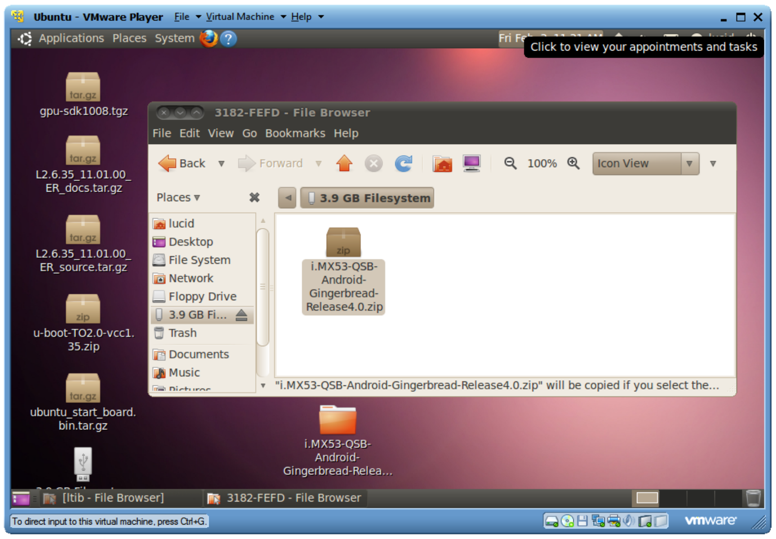
13. i.MX53 ("http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=IMX53QSB") preparation as an Android device. The following steps apply only to i.MX53 development board by Freescale and need to be done only once for each board:

    - i.MX53 development board is out the box VGA ready. In order to use LCD instead, connect i.MX53's RS-232 port to host PC (in case PC is not equipped with RS-232 port, use RS-232/USB concverter).
    - Launch "Hyper Terminal" or other program to communicate via serial port, power up i.MX53 board and press any key when "AUTOBOOT WILL BEGIN IN:" appears.
    - After the prompt, type: "set bootargs_base 'set bootargs console=ttymxc0,115200 ${lcd}'"
    - Press "Enter".
    - Type "saveenv" and press "Enter" to save changes on the microSD card.
    - Type "boot" and press "Enter" to continue boot up procedure.

14. i.MX53 comes with Linux UBUNTU on SD card. To use Android OS, BSP (Board Support Package) from Adeneo "http://www.adeneo-embedded.com/iMX53" needs to be used to copy Android Image onto SD card that would be used to boot up i.MX53 from. If Android OS was already running on the target device this step would not be required.

    Per "http://www.adeneo-embedded.com/iMX53": "Adeneo Embedded ported Android and Windows Embedded Compact 7 operating systems into the Freescale i.MX53 Quick Start board (QSb), a low cost development platform ($149 only) based on an ARM® Cortex-A8 1 GHz processor. The i.MX53 QSb includes a display controller, hardware-accelerated graphics, 1080p video decode and 720p encode as well as numerous connectivity options ideally suited for applications such as human machine interface in embedded consumer, industrial and medical markets."

- Download VMware player from "http://www.vmware.com/products/player/" and install it on the host PC. Use default options. This will be used to prepare SD card with Android OS in Linux Ubuntu environment (in case of using Windows OS to create SD card with bootable Android OS).
- Restart host computer.
- Copy "VM_UBUNTU.zip" from the DVD that came with i.MX53 board and extract it to desired directory on host PC.
- Launch VMware through double clicking "Ubuntu.vmx" file under ".../vm_Ubuntu" directory.
- When "VMware Player" asks whether the virtual computer was "moved" or "copied", choose "copied".
- Log into the virtual machine with the following credentials:
    - username= lucid
    - password= lucid
- Download i.MX53_Android Source BSP from "http://www.adeneo-embedded.com/iMX53" (this will be installed on SD card that i.MX53 will later use to boot from). In case of downloading it under Windows, instead of under Ubuntu, Store it temporarily on the 4GB SD card after downloading it, in order to copy it from that SD card onto a storage device (harddrive) managed by Ubuntu accessed via Vmware later.
- If i.MX53_Android Source BSP (for this project i.MX53-QSB-Android-Gingerbread-Release4.0.zip) was downloaded under Ubuntu and stored on the storage device accessible via Ubuntu skip next step, otherwise:
- Log into Ubuntu. Insert previously prepared SD card to SD card reader or to USB port using SD/USB converter.
- Under Ubuntu, extract i.MX53-QSB-Android-Gingerbread-Release4.0.zip from SD card onto the hard drive and extract to desired directory as shown in Fig. 86.

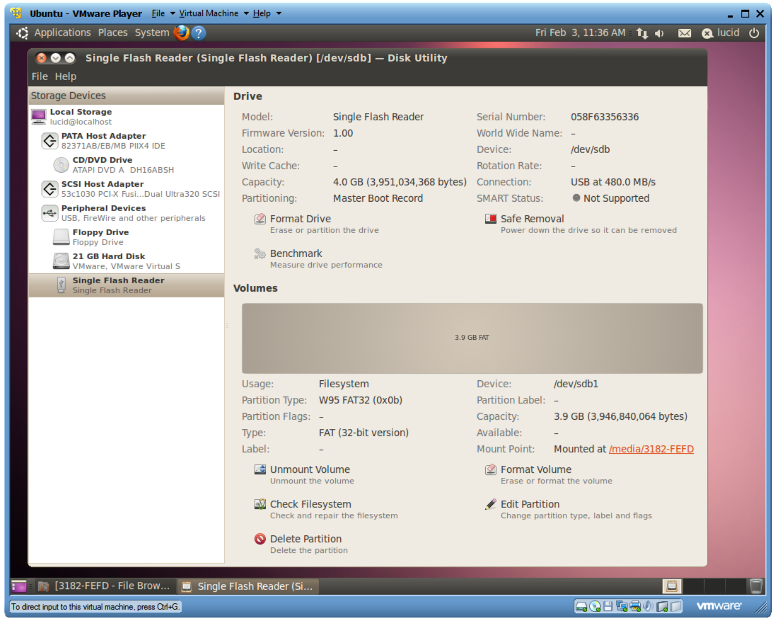


"i.MX53-QSB-Android-Gingerbread-Release4.0.zip" will be copied if you select the...

To direct input to this virtual machine, press Ctrl+G.

Fig. 86) i.MX53-QSB-Android-Gingerbread-Release4.0 extraction

- Learn the Linux name of the SD card storage device. Use "Disk Utility" to verify it, as shown in Fig. 87.


Single Flash Reader (Single Flash Reader) [/dev/sdb] — Disk Utility

Storage Devices
- Local Storage — lucid@localhost
- PATA Host Adapter — 82371AB/EB/MB PIIX4 IDE
- CD/DVD Drive — ATAPI DVD A DH16ABSH
- SCSI Host Adapter — 53c1030 PCI-X Fusi...Dual Ultra320 SCSI
- Peripheral Devices — USB, FireWire and other peripherals
- Floppy Drive — Floppy Drive
- 21 GB Hard Disk — VMware, VMware Virtual S
- Single Flash Reader — Single Flash Reader

Drive

Model:	Single Flash Reader	Serial Number:	058F63356336
Firmware Version:	1.00	World Wide Name:	–
Location:	–	Device:	/dev/sdb
Write Cache:	–	Rotation Rate:	–
Capacity:	4.0 GB (3,951,034,368 bytes)	Connection:	USB at 480.0 MB/s
Partitioning:	Master Boot Record	SMART Status:	● Not Supported

Volumes

3.9 GB FAT

Usage:	Filesystem	Device:	/dev/sdb1
Partition Type:	W95 FAT32 (0x0b)	Partition Label:	–
Partition Flags:	–	Capacity:	3.9 GB (3,946,840,064 bytes)
Type:	FAT (32-bit version)	Available:	–
Label:	–	Mount Point:	Mounted at /media/3182-FEFD

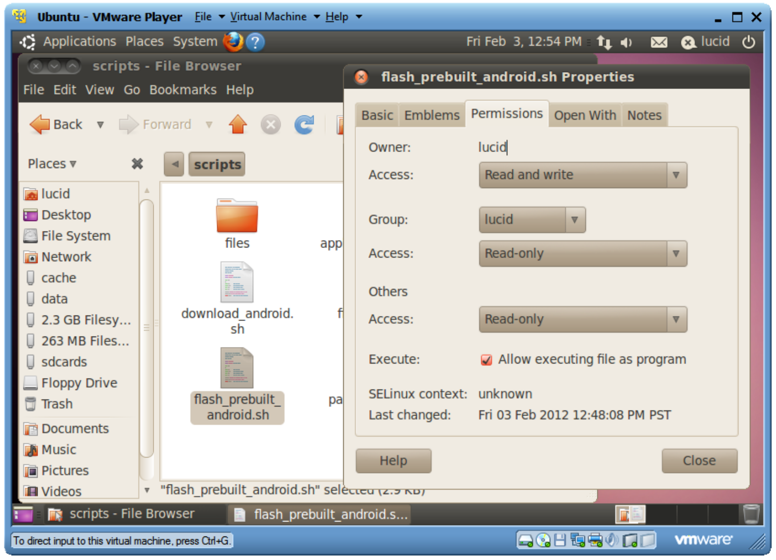
To direct input to this virtual machine, press Ctrl+G.

Ubuntu - VMware Player
File
Virtual Machine
Help
Applications Places System
Fri Feb 3, 11:36 AM
lucid
File Help
Format Drive
Erase or partition the drive
Safe Removal
Power down the drive so it can be removed
Benchmark
Measure drive performance
Unmount Volume
Unmount the volume
Format Volume
Erase or format the volume
Check Filesystem
Check and repair the filesystem
Edit Partition
Change partition type, label and flags
Delete Partition
Delete the partition
[3182-FEFD - File Brow...
Single Flash Reader (Si...
vmware

Fig. 87) SD (Secure Digital) as a Linux device

In this example, the name is: "/dev/sdb".

- Find "flash_prebuilt_android.sh" script under "i.MX53-QSB-Android-Gingerbread-Release4.0/scripts/"
- Right click on "flash_prebuilt_android.sh", navigate to "Permissions" and check "Allow executing file as program" as shown in Fig. 88:


flash_prebuilt_android.sh Properties

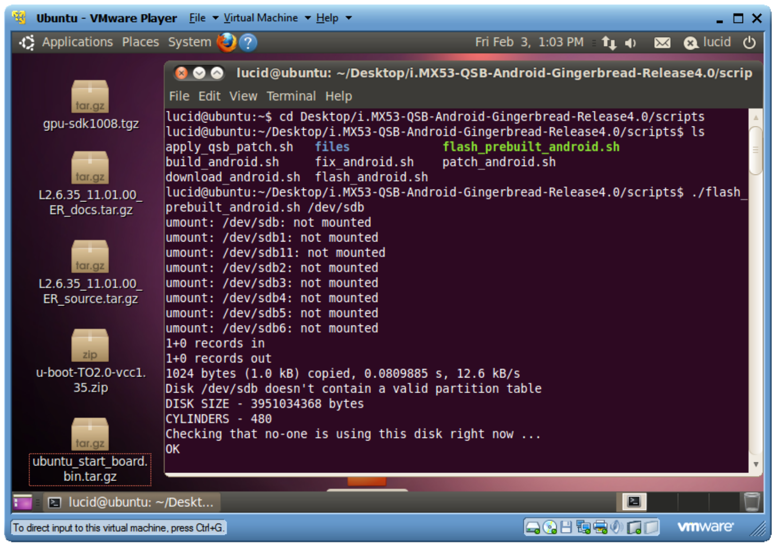
Owner:	lucid
Access:	Read and write
Group:	lucid
Access:	Read-only
Others Access:	Read-only
Execute:	[✓] Allow executing file as program
SELinux context:	unknown
Last changed:	Fri 03 Feb 2012 12:48:08 PM PST

"flash_prebuilt_android.sh" selected (2.9 KB)

To direct input to this virtual machine, press Ctrl+G.

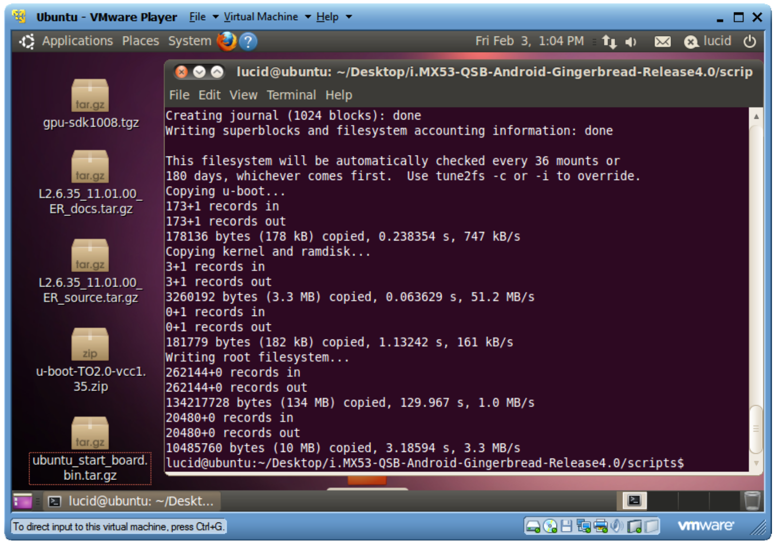
Fig. 88) "flash_prebuilt_android.sh" script permission setup

- Launch "Terminal".
- Type "cd Desktop/i.MX53-QSB-Android-Gingerbread-Release4.0/scripts". Type ls and hit enter to check the contents of the directory (look for "flash_prebuilt_android.sh").
- Type "./flash_prebuilt_android.sh /dev/sdb", which will format and install Android OS on the SD card that will be used by i.MX53 device later. Fig. 89-90 show the process' progress.


Ubuntu - VMware Player
File
Virtual Machine
Help
Applications
Places
System
Fri Feb 3, 1:03 PM
lucid
lucid@ubuntu: ~/Desktop/i.MX53-QSB-Android-Gingerbread-Release4.0/scrip
File Edit View Terminal Help
gpu-sdk1008.tgz
L2.6.35_11.01.00_ER_docs.tar.gz
L2.6.35_11.01.00_ER_source.tar.gz
u-boot-TO2.0-vcc1.35.zip
ubuntu_start_board.bin.tar.gz
lucid@ubuntu: ~/Deskt...
To direct input to this virtual machine, press Ctrl+G.

```
lucid@ubuntu:~$ cd Desktop/i.MX53-QSB-Android-Gingerbread-Release4.0/scripts
lucid@ubuntu:~/Desktop/i.MX53-QSB-Android-Gingerbread-Release4.0/scripts$ ls
apply_qsb_patch.sh   files              flash_prebuilt_android.sh
build_android.sh     fix_android.sh     patch_android.sh
download_android.sh  flash_android.sh
lucid@ubuntu:~/Desktop/i.MX53-QSB-Android-Gingerbread-Release4.0/scripts$ ./flash_
prebuilt_android.sh /dev/sdb
umount: /dev/sdb: not mounted
umount: /dev/sdb1: not mounted
umount: /dev/sdb11: not mounted
umount: /dev/sdb2: not mounted
umount: /dev/sdb3: not mounted
umount: /dev/sdb4: not mounted
umount: /dev/sdb5: not mounted
umount: /dev/sdb6: not mounted
1+0 records in
1+0 records out
1024 bytes (1.0 kB) copied, 0.0809885 s, 12.6 kB/s
Disk /dev/sdb doesn't contain a valid partition table
DISK SIZE - 3951034368 bytes
CYLINDERS - 480
Checking that no-one is using this disk right now ...
OK
```

Fig. 89) "flash_prebuilt_android.sh" script execution


Ubuntu - VMware Player
File
Virtual Machine
Help
Applications
Places
System
Fri Feb 3, 1:04 PM
lucid
lucid@ubuntu: ~/Desktop/i.MX53-QSB-Android-Gingerbread-Release4.0/scrip
File Edit View Terminal Help
gpu-sdk1008.tgz
L2.6.35_11.01.00_ER_docs.tar.gz
L2.6.35_11.01.00_ER_source.tar.gz
u-boot-TO2.0-vcc1.35.zip
ubuntu_start_board.bin.tar.gz
lucid@ubuntu: ~/Deskt...
To direct input to this virtual machine, press Ctrl+G.

```
Creating journal (1024 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 36 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
Copying u-boot...
173+1 records in
173+1 records out
178136 bytes (178 kB) copied, 0.238354 s, 747 kB/s
Copying kernel and ramdisk...
3+1 records in
3+1 records out
3260192 bytes (3.3 MB) copied, 0.063629 s, 51.2 MB/s
0+1 records in
0+1 records out
181779 bytes (182 kB) copied, 1.13242 s, 161 kB/s
Writing root filesystem...
262144+0 records in
262144+0 records out
134217728 bytes (134 MB) copied, 129.967 s, 1.0 MB/s
20480+0 records in
20480+0 records out
10485760 bytes (10 MB) copied, 3.18594 s, 3.3 MB/s
lucid@ubuntu:~/Desktop/i.MX53-QSB-Android-Gingerbread-Release4.0/scripts$
```

Fig. 90) "flash_prebuilt_android.sh" script execution progress

- SD card is ready to be inserted into i.MX53 (mini SD card slot). Android Gingerbread OS will be launched after i.MX53 is powered up.

15. Blender installation.

16. Download Blender version 2.49 (collada file exported by Blender v2.49 was tested with Inflexion UI PC tool used for this project). In the future newer and more user friendly revisions of Blender might possibly become compatible with further revisions of Inflexion UI.

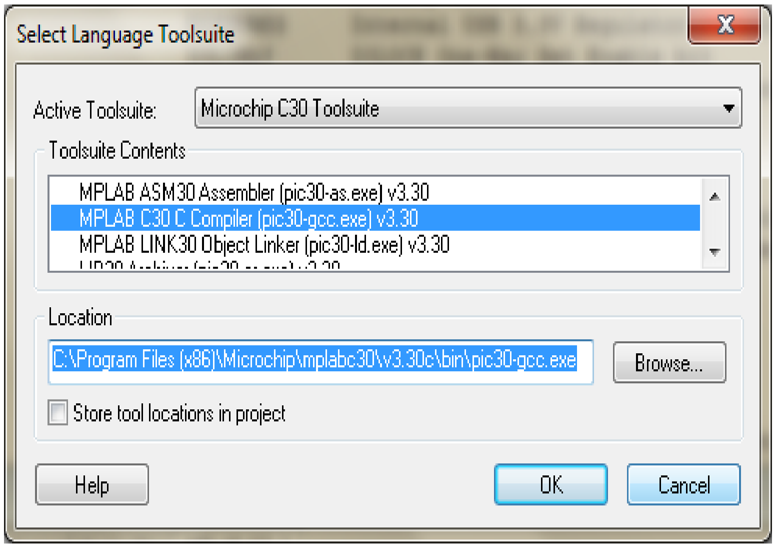
    - Download "Blender-2.49-win64.zip" for Windows7(x64), or pick a suitable one for other OS, from "http://download.blender.org/release/Blender2.49/"
    - No installation is necessary. Simply launch "blender.exe" located in extracted directory in order to create 3D objects for the design.

17. Target Android Device (i.MX53) is now ready to launch Android Applications with user interface/graphics developed using Inflexion PC tool and integrated with Android application code (JAVA, C/C++) in Eclipse IDE. Developer needs to launch Inflexion UI, develop desired user interface using imported (custom) 3D object (developed in Blender and exported into Collada format) and/or objects available through Inflexion library. Then he needs to integrade Inflexion output files with Android application developed in Eclipse and load it onto Target device using Eclipse IDE features.

18. Download MPLAB, a graphical, integrated debugging tool set for all of Microchip's: 8-bit, 16-bit and 32-bit MCUs digital signal controllers, and memory devices, from "http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002" and install using default settings. Microchip development environment is needed to built C code to handle communication between MPU-6000 (I2C) and Android application (USB). Moreover, the C code implements MotionFusion algorithms.

19. Download MPLAB C Compiler for Academic Use from "http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en536656 " and install on the host machine using default settings.

    - Navigate to "Project/Select Language Toolsuite" and pick "MPLAB C30 Toolsuite" as shown in Fig. 91. Click "OK".



Select Language Toolsuite

Active Toolsuite: Microchip C30 Toolsuite

Toolsuite Contents

MPLAB ASM30 Assembler (pic30-as.exe) v3.30
MPLAB C30 C Compiler (pic30-gcc.exe) v3.30
MPLAB LINK30 Object Linker (pic30-ld.exe) v3.30

Location

C:\Program Files (x86)\Microchip\mplabc30\v3.30c\bin\pic30-gcc.exe

Store tool locations in project

Fig. 91) MPLAB Toolsuite setup

20. Plug in the programmer into USB port of the host PC in order to update the microcontroller with the firmware built using C30 compiler (for this project ICD3 programmer has been used). "Plug and Play" installation will be performed by Microchip tool and programmer will immediately become available afterwards.

Appendix B

Schematic & PCB

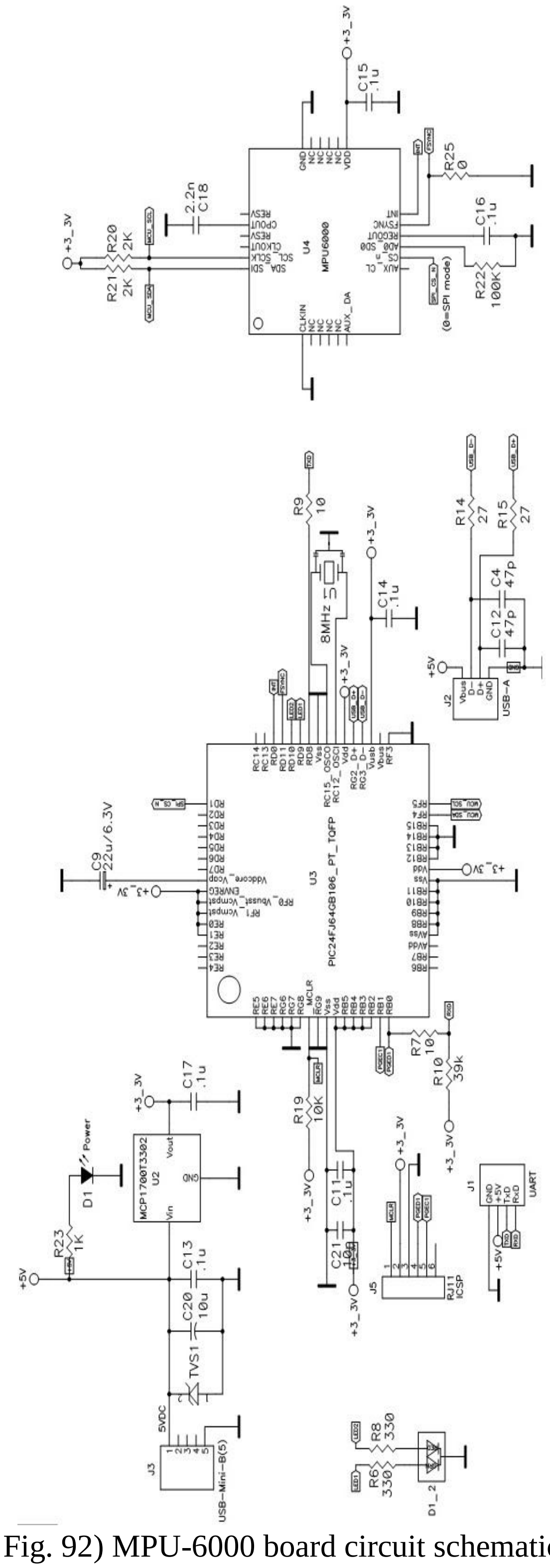


Fig. 92) MPU-6000 board circuit schematic

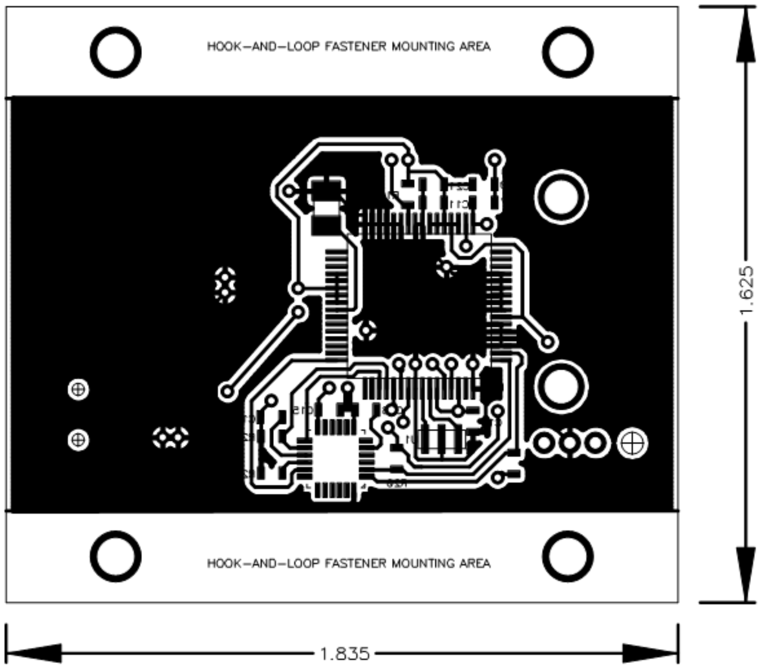

HOOK–AND–LOOP FASTENER MOUNTING AREA
HOOK–AND–LOOP FASTENER MOUNTING AREA
1.625
1.835

Fig. 93) MPU-6000 PCB: Bottom side

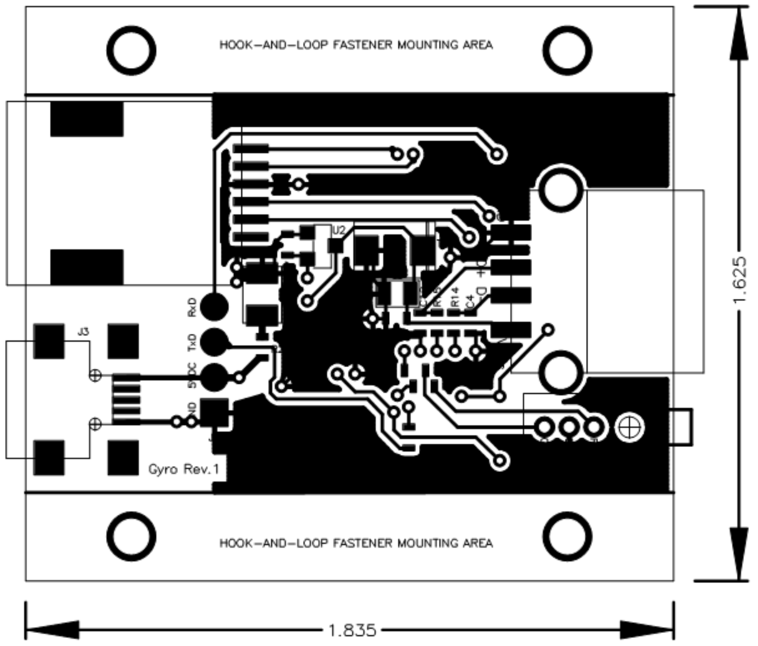

HOOK–AND–LOOP FASTENER MOUNTING AREA
U2
R14
C4
RxD
TxD
5VDC
GND
J3
Gyro Rev.1
HOOK–AND–LOOP FASTENER MOUNTING AREA
1.625
1.835

Fig. 94) MPU-6000 PCB: Top side

Attached in an electronic form.

Appendix C

Bill of Materials

Item	Quantity	Reference	Part	PCB Footprint
1	1	D1_2	DUAL COLOR LED	thruhole
2	6	C11	.1u	0603
3		C13		
4		C14		
5		C15		
6		C16		
7		C17		
8	1	C18	2.2n	0603
9	1	C21	10n	0603
10	2	C4	47p	0603
11		C12		
12	1	C20	10u	0805
13	1	C9	22u/6.3V	SMD
14	1	D1		SMD
15	1	U2	MCP1700T3302	SOT-23
16	1	U4	MPU-6000	
17	1	U3	PIC24FJ256GB106	
18	1	R25	0	0603
19	1	R23	1k	0603
20	2	R20	2k	0603
21		R21		
22	2	R7	10	0603
23		R9		
24	1	R19	10k	0603
25	2	R14	27	0603
26		R15		
27	1	R10	39k	0603
28	1	R22	100k	0603
29	2	R6	330	0603
30		R8		
31	1	U1	RESONATOR 8MHz	
32	1	J5	ICSP	RJ11-SMD
33	1	TVS	6V	SMD
34	1	J1	UART-PADS	
35	1	J3	USB-Mini-B	SMD

36	1	J2	USB-A	SMD

Fig. 95) BOM

Appendix D

Android Application Source Code

Attached in an electronic form.

Appendix E

Inflexion Source

Attached in an electronic form.

Appendix F

Firmware Source Code

Attached in an electronic form.