PROGRESS REPORT

OF PROJECT

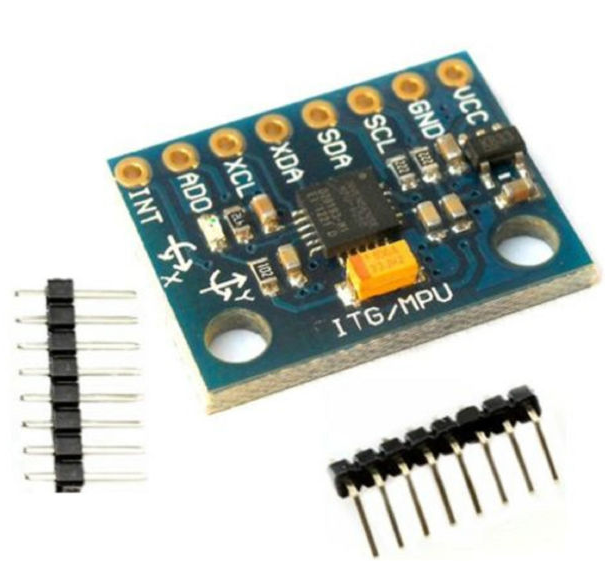
AIRSENS

**STATUS**

For the last one month we have been doing research for both hardware and software needed to build the project. We have found the appropriate requirements of the project along with the verification of the feasibility of the project.

**HARDWARE REQUIREMENTS**

1. 2-AXIS ACCELEROMETER + GYROMETER CHIP:

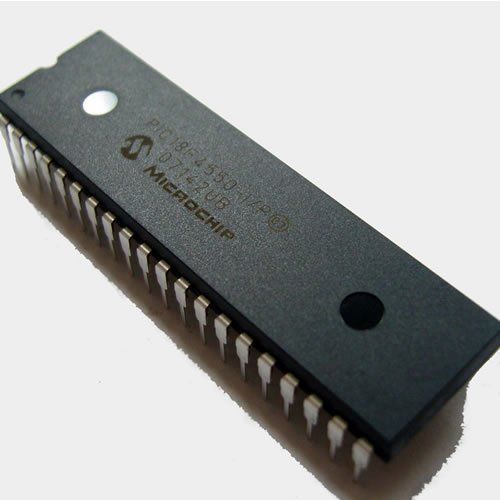


2X MPU-6050 MODULE GY-521 + 2 x sets of header pins

**Product description:**

* MPU-6050 module (Three Axis Gyro + Three Axis Acceleration)
* Chip used: MPU-6050
* Power supply: 3-5 v (internal low voltage difference voltage stabilizing)
* Communication: standard IIC communication agreement
* Chip In-built 16 bit AD converter, 16 bits of data output
* The gyroscope range: + 250 500 1000 2000 degree /sec
* Acceleration range: ± 2g ± 4g ± 8g ± 16 g
* Net weight:3 g
* Package weight:13 g
* The MPU-6050 is a high precision piece of motion processing tech.
* By combining a MEMS 3-axis gyroscope and a 3-axis accelerometer on the same silicon die together with an on board Digital Motion Processor™ (DMP™) capable of processing complex 9-axis Motion Fusion algorithms
* MPU-6050 does away with the cross-axis alignment problems that can creep up on discrete parts.
* Auxiliary master I2C bus which allows the MPU-6050 to access external magnetometers and other sensors.

1. PIC MICROCONTROLER:



**Features:**

Full Speed USB 2.0 (12Mbit/s) interface

1K byte Dual Port RAM + 1K byte GP RAM

Full Speed Transceiver

16 Endpoints (IN/OUT)

Internal Pull Up resistors (D+/D-)

48 MHz performance (12 MIPS)

Pin-to-pin compatible with PIC16C7X5

|  |  |
| --- | --- |
| **Parameter Name** | **Value** |
| Program Memory Type | Flash |
| Program Memory (KB) | 32 |
| CPU Speed (MIPS) | 12 |
| RAM Bytes | 2,048 |
| Data EEPROM (bytes) | 256 |
| Timers | 1 x 8-bit, 3 x 16-bit |
| ADC | 10 ch, 10-bit |
| Comparators | 2 |
| USB (ch, speed, compliance) | 1, FS Device, USB 2.0 |
| Temperature Range (C) | -40 to 85 |
| Operating Voltage Range (V) | 2 to 5.5 |
| Pin Count | 28 |

**SOFTWARE REQUIREMENTS**

1. Embedded C:

**Embedded C** is a set of language extensions for the C Programming language by the C Standards committee to address commonality issues that exist between C extensions for different embedded systems. Historically, embedded C programming requires nonstandard extensions to the C language in order to support exotic features such as fixed-point arithmetic, multiple distinct memory banks, and basic I/O operations. In 2008, the C Standards Committee extended the C language to address these issues by providing a common standard for all implementations to adhere to. It includes a number of features not available in normal C, such as, fixed-point arithmetic, named address spaces, and basic I/O hardware addressing.

Embedded C uses most of the syntax and semantics of standard C, e.g., main() function, variable definition, data type declaration, conditional statements (if, switch, case), loops (while, for), functions, arrays and strings, structures and union, bit operations, macros, etc.A Technical Report was published in 2004 and a second revision in 2006.

Necessity: During infancy years of microprocessor based systems, programs were developed using assemblers and fused into the EPROMs. There used to be no mechanism to find what the program was doing. LEDs, switches, etc. were used to check for correct execution of the program. Some ‘very fortunate’ developers had In-circuit Simulators (ICEs), but they were too costly and were not quite reliable as well. As time progressed, use of microprocessor-specific assembly-only as the programming language reduced and embedded systems moved onto C as the embedded programming language of choice. C is the most widely used programming language for embedded processors/controllers. Assembly is also used but mainly to implement those portions of the code where very high timing accuracy, code size efficiency, etc. are prime requirements.As assembly language programs are specific to a processor, assembly language didn’t offer portability across systems. To overcome this disadvantage, several high level languages, including C, came up. Some other languages like PLM, Modula-2, Pascal, etc. also came but couldn’t find wide acceptance. Amongst those, C got wide acceptance for not only embedded systems, but also for desktop applications.

Even though C might have lost its sheen as mainstream language for general purpose applications, it still is having a strong-hold in embedded programming. Due to the wide acceptance of C in the embedded systems, various kinds of support tools like compilers & cross-compilers, ICE, etc. came up and all this facilitated development of embedded systems using C. Assembly language seems to be an obvious choice for programming embedded devices. However, use of assembly language is restricted to developing efficient codes in terms of size and speed. Also, assembly codes lead to higher software development costs and code portability is not there. Developing small codes are not much of a problem, but large programs/projects become increasingly difficult to manage in assembly language. Finding good assembly programmers has also become difficult nowadays. Hence high level languages are preferred for embedded systems programming.

Advantages

* It is small and simpler to learn, understand, program and debug.
* Compared to assembly language, C code written is more reliable and scalable, more portable between different platforms.
* C compilers are available for almost all embedded devices in use today, and there is a large pool of experienced C programmers.
* Unlike assembly, C has advantage of processor-independence and is not specific to any particular microprocessor/microcontroller or any system. This makes it convenient for a user to develop programs that can run on most of the systems.
* As C combines functionality of assembly language and features of high level languages, C is treated as a ‘middle-level computer language’ or ‘high level assembly language’.
* It is fairly efficient.
* It supports access to I/O and provides ease of management of large embedded projects.
* Java is also used in many embedded systems but Java programs require the Java Virtual Machine (JVM), which consumes a lot of resources. Hence it is not used for smaller embedded devices.

1. JAVA:

**Java** is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2015, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licences. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Class path (standard libraries), and Iced Tea-Web (browser plugin for applets).

The latest version is Java 8, the only supported version, currently.

**ACCOMPLISHMENTS**

1. FEASIBILITY OF THE PROJECT:

Our main task was to find whether this project is feasible or not. Is it possible to map the hand movement using accelerometer on the computer screen. To verify this we used the accelerometer of the smart-phones that are used widely. We made an app that transmits the readings of the accelerometer of the phone to the computer.

We also made a computer application that after receiving the data from the phone i.e. X & Y accelerations depicts the proper reception of the data and also some action to verify that we can use the accelerations to represent movements.

This is the main concept that was doubtful at the beginning but is verified now.

1. OTHER:

We also found that in java a class called **"robot"** can be used to write the mouse driver that we intend to make for the project. The main technologies and requirements for both hardware and software has been researched and decided for the use in the project.

Defining these requirements helps us to confine our work on limited things rather doing hit and trial to find the perfect match.

**TO ACCOMPLISH**

1. The main task is to find an algorithm to map the virtual plane of the airsens to the computer screen.
2. To fabricate the accelerometer and gyro-meter chip and also researching about connecting units required between accelerometer chip, PIC Microcontroller and the computer.
3. To deploy the finalised product AIRSENS after thorough testing and evaluation.