



Czech Technical University in Prague

FACULTY OF BIOMEDICAL ENGINEERING

Department of Biomedical Technology

The Posture Evaluation System

Bachelor Thesis

Study program: Biomedical and Clinical Technology
Study branch: Biomedical Technician

Author of thesis: Cornil Petrov
Supervisor of thesis: Ing. Tomáš Funda

Kladno, May 2018

Department of Biomedical Technology

Academic year: 2017/2018

Bachelor thesis assignment

Student: **Cornil Igorevich Petrov**
Study branch: Biomedical Technician
Title: **The posture evaluation system**
Title in Czech: Systém pro hodnocení držení těla

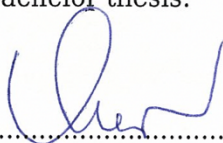
Instructions for processing:

Design a prototype of posture evaluation circuit. Use commercially available sensors, i.e. gyroscope and accelerometer. The signal from the sensors process by microprocessor of Arduino type or Raspberry Pi. Test the designed equipment for artificial conditions (outside the human body) and on the human body as well. Write a technical documentation of designed solution.

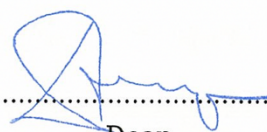
List of Literature:

- [1] OHYA, Jun, Akira. UTSUMI a Junji. YAMATO, Analyzing video sequences of multiple humans: tracking, posture estimation and behavior recognition, Boston: Kluwer, 2002, ISBN 1-4020-7021-7
- [2] Jack M. WINTERS, Patrick E. CRAGO (eds.), Biomechanics and neural control of posture and movement, New York: Springer, 2000, ISBN 0-387-94974-7
- [3] Smith, A., Introduction to Arduino: A Piece of Cake, 14.5.2013, [Revidováno 16.10.2015], [Citováno 07.11.2017], <https://www.introtoarduino.com/>
- [4] Raspberry Pi Foundation, Raspberry Pi, 13.8.2014, [Revidováno 18.1.2016], [Citováno 07.11.2017], <https://www.raspberrypi.org/>

Validity of assignment until date: 20.09.2019
Supervisor of bachelor thesis: Ing. Tomáš Funda
Consultant of bachelor thesis: MUDr. Markéta Janatová



Head of Department



Dean

In Kladno, 19.02.2018

DECLARATION

I hereby declare that I have completed this thesis with the topic “The posture evaluation system” independently and I have included a full list of used references.

I do not have a compelling reason against the use of the thesis within the meaning of Section 60 of the Act No 121/2000 Coll., on copyright, rights related to copyright and amending some laws (Copyright Act).

In Kladno, date 18.05.2018

.....

ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to Ing. Tomáš Funda for providing me with all the tools and basic skills necessary for the construction of this device.

Ing. Tomáš Funda's consistent encouragement, guidance and patience enabled me to complete this project successfully.

I would also like to thank Ing. Petr Kudrna, Ph.D. and MUDr. Markéta Janatová for their advisory throughout the thesis.

ABSTRACT

Thesis Title: The Posture Evaluation System

Proper posture does not only display confidence and stature, but is a sign of well-being in general. For healthy people, maintaining improper posture poses many health risks, including spinal misalignment, deformation of musculoskeletal structures and pain in general. In today's computer age, when people have no motivation to align their axes properly, it's more important than ever to maintain proper posture. The suggested project allows a posture evaluation device to be constructed from commercially available devices, which sends signals from the inclining object to the LCD display and speaker, warning the user of a posture/inclination fault. Responsible for the precise measurements is the MEMS gyroscope in the 6-axis motion sensor connected to the circuit. All calibration and test processes were described to achieve optimal performance in detecting posture fault

Keywords: Posture, gyroscope, Arduino, micro-controller, HD6050, fault detection

Content

List of symbols and abbreviations	8
1 Introduction	9
1.1 Steady state	10
1.2 Aim	10
2 Design.....	11
2.1 Inter-Integrated Circuit	12
2.2 Microcontroller	12
2.3 Gyroscope sensor.....	12
2.4 LCD Display	13
2.5 Piezoelectric speaker	13
2.6 External Button.....	13
2.7 9V Battery and Switch.....	13
2.8 Plastic Cover.....	14
3 Devices and Elements Used	15
3.1 Arduino Leonardo.....	16
3.2 InvenSense 6-axis MPU-6050 Gyroscope-Accelerometer.....	18
3.3 LCD HD44780.....	21
3.4 Piezoelectric Speaker.....	24
3.5 9v Battery and Switch.....	25
3.6 External Button.....	26
3.7 Plastic Casing.....	27
4 Results	30
5 Calibration and Testing	33
5.1 Calibration and Verification	33
5.2 Human Subject Testing.....	36
5.2.1 Posture Evaluation Session	37
5.2.2 Questionnaire/Form	39
6 Testing Results	40
7 Discussion	41
8 Conclusion.....	45
References.....	46

Appendix A. Main code for posture evaluation.....	48
Appendix B Code for retrieving RAW values.....	52
Appendix C I ² C scanner.....	55
Appendix D code for processing DMP offsets	56
Appendix E Testing Questionnaire	58

List of Symbols and Abbreviations

Abbreviation	Meaning
MC	Micro-controller
IDE	Integrated Development Environment
DMP	Digital Motion Processor
LCD	Liquid Crystal Display
FIFO	First in first out
MEMS	Microelectromechanical systems
ICSP	In-circuit serial programming
PWM	Pulse-width Modulation
I/O	Input/Output

Symbol	Meaning
I ² C	Inter-Integrated Circuit
SCL	I ² C bus clock connection
SDA	I ² C bus data lines connection
VDD	Integrated circuit power supply

1 Introduction

Anatomical posture in humans is controlled by the cerebellum and motor cortex of the brain. Rigidity is maintained by slow twitch fibers in the muscles. These fibers are more efficient in generating ATP through aerobic processes resulting in extended muscle contractions over longer periods of time. [1] A good posture implies a minimum expenditure of energy required to maintain proper anatomical alignment, whereas exertion of excess energy and effort is a sign of poor posture. Joints are the most mechanically efficient while maintaining good posture, which provides the body with ease of movement for less effort. In a good posture the weight is equally distributed, all axes are parallel to an arbitrary vertical line, shoulders are in an erect position with chest held up, abdomen is retracted back and curves of the spine are not twisted. [2]

Improper posture exerts unbalanced forces on the supporting structures of the body: spinal column and discs, pelvis, joints, muscles and tendons. This leads to fatigue, appearance abnormalities, deformation, pain, difficulty of movement and balance. [3]

Causes of a poor posture can be either acquired or congenital. Genetics, disease, habit, physical trauma and weakness all affect posture. [2] An individual suffering from Parkinson's disease experiences a forward flex posture, hindering normal functionality. In healthy individuals, habit plays a vital role in tendency to maintain proper posture. People tend to remain in a poor posture, regardless of their desire to maintain proper posture. This is usually attributed poor habit. These may be formed at an early age from improper initial development or acquired later in life as a result of a leading sedentary lifestyle. As a consequence of technological industrialization, work with computers is becoming extremely common, resulting in an increase of individuals working in a sedentary posture for long periods of time. [4] In addition to excessive use of computers, other harmful habits, such as improper form during exercise, carrying heavy loads and sitting behind a desk for long periods of time affects the shape of muscles and the skeletal development of an individual. This may cause abnormal development leading to difficulty maintaining proper posture. [5]

Habits are automatic responses to contextual cues, or signals, associated with their execution. For example, putting on a seat belt when getting into a car. A habitual action is dependent mostly on external cues without requiring conscious attention or motivational processes. Habits are therefore likely to persist even without conscious

awareness of the action. Formation of habits is simply constituted by repeating an action consistently in the same context. [6,7]

1.1 Steady state

When it comes to maintaining proper spinal alignment there are many braces available designed to perform this function. However, this does not facilitate muscle strengthening, which ultimately has little impact on body posture.

Since the mid-2000s, many posture correction apparatuses based on habit formation were patented and have come to be available on the market. However, few have achieved commercial success and popular appraisal. [8] “LumoMotionScience” and “Upright” have been the leaders on the market for the past several years. The products are portable and contain a sensor inside the device. The sensor transmits positional data to the user’s mobile application via Bluetooth adapter, providing various information on body position. The device vibrates when slouching is detected, warning the user of posture fault.

However, these solutions do not display deviation values in real time and require a connection to a smartphone with Bluetooth capabilities. This may not be a suitable option in certain clinical or rehabilitation scenarios.

1.2 Aim

The aim of this project is to design and test a prototype of a posture evaluation prototype using commercially available devices from which the signals are processed by an Arduino microcontroller. The prototype will allow the user to observe misalignment of the upper back as a percent value of deviation from the vertical axis during pitch and yaw movements. Visual and audio signals will serve as repeated external cues warning the user of deviation. Sensors are to be tested and calibrated both outside and on the human body.

The hypothesis is that it is possible to construct a portable posture evaluation device that would dynamically display vertical deviation values and promote healthy postural habit formation, without requiring a connection to mobile applications.

2 Design

Four primary steps were required for achieving a successful and clinically viable result in the realization of detecting posture fault on a human subject:

1. Connection of components
2. Protection/Fixation
3. Sensor calibration
4. Testing/Implementation

A block scheme of the purposed solution is presented in Figure 2.1.

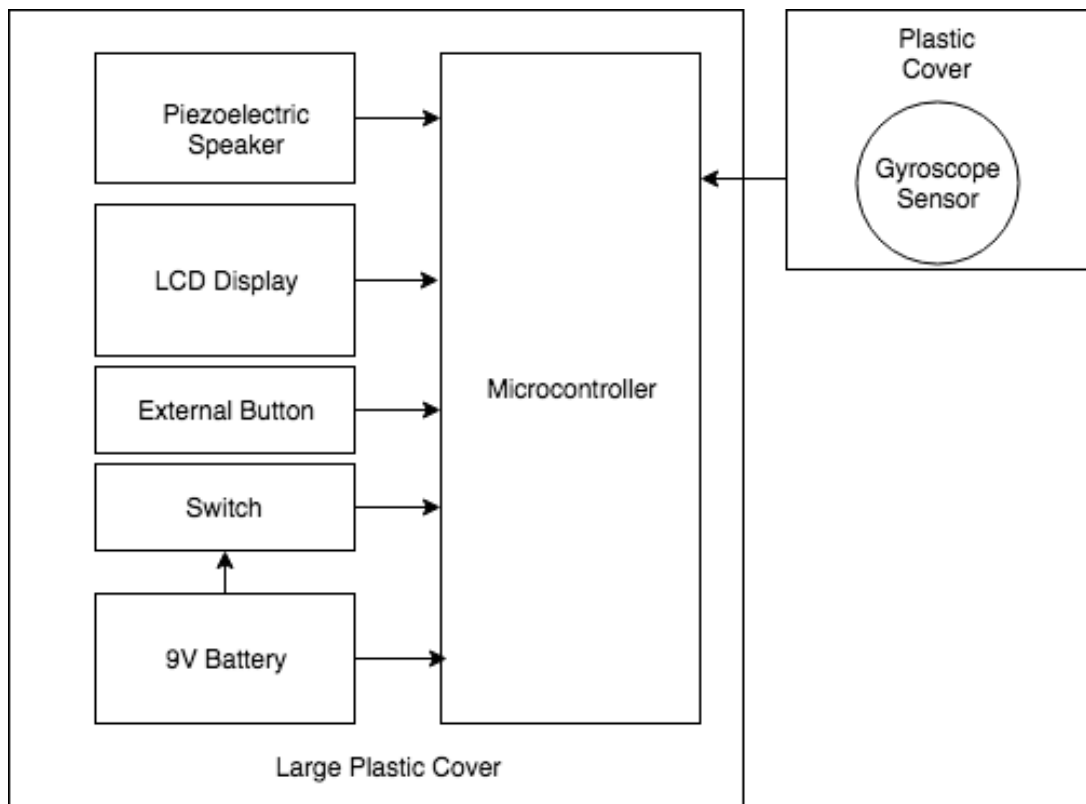


Figure 2.1. Block scheme of posture evaluation device

2.1 Inter-Integrated Circuit

Communication of devices should be performed via Inter-Integrated Circuit (I²C), a serial bus commonly used for attachment of lower speed peripherals to microcontrollers. It uses two lines as a collector and drain: Serial Data Line (SDA) and Serial Clock Line (SCL) with typical voltages 3,3-5V. Its simplicity and accessibility makes it the ideal communications protocol for this device, as all components connected to the bus do not exceed 5V range, while SDA and SCL lines with I²C communication are commonly found on most microcontrollers.

2.2 Microcontroller

The microcontroller is responsible for providing means for connecting all the elements of the posture evaluation system and processing all information that is exchanged between them. A microcontroller contains a CPU along with memory and programmable digital and analog input/output peripherals. It is mainly used for executing embedded applications uploaded to the memory. Its' size and cost efficiency makes it ideal for implementation in this device.

2.3 Gyroscope sensor

The gyroscope sensor sends motion data to the microcontroller for further processing. It should measure in 3-axis, 2 of which would be used in context for the realization of angle detection within the prototype. The gyroscope sensor should also be of microelectromechanical system (MEMS) type, meaning there is a piezoelectric component that vibrates and sends information about movement to the computer as an electrical signal. The sensor should be able to communicate with I²C protocol as well as process and convert analog to digital signal to the microcontroller. It is necessary to install software libraries and perform calibration of the sensor.

2.4 LCD Display

The LCD is responsible for displaying text and current deviation from data collected from the gyroscope sensor, visually alarming the user when improper posture is detected while displaying deviation values. The display should be 5V maximum and have I²C interface for ease of communication with the MC. Most libraries for LCD character display may be downloaded online. Small size and cost efficiency makes a 16 pin LCD display of smaller size ideal for implementation in this device.

2.5 Piezoelectric speaker

The piezoelectric speaker is responsible for alarming the user of detected posture fault. As electricity flows through the speaker, it goes through a ceramic material with piezoelectric properties. The voltage applied by the current produces vibrations. The vibrations result in a noise signal, triggered by high values of deviation from information processed by the MC. Changing of the noise is directly influenced by the frequency of the input signal, which is set by the user in microcontroller development environment.

2.6 External Button

A simple push button is intended to initiate the posture evaluation process in the device. When the push button is unpressed, the pin is connected to HIGH voltage state. Pressing the button makes a connection between its legs connecting the pin to ground creating LOW pin voltage state. The gyroscope data collection should be initiated by the button pin state LOW.

2.7 9V Battery and Switch

A regular 9V battery with connected switch are responsible for providing power to the prototype. The battery is crucial for portability, as it can be easily placed within the device and powered “ON” and “OFF” by the user using the connected switch. The switch works similar to the external button, completing the circuit between the power source and the MC.

2.8 Plastic Cover

The plastic covers for the device and sensor are responsible for maintaining the integrity of the devices inside, as well as providing fixation and protection. The plastic case used should have dimensions capable of fitting all devices inside, excluding the sensor. The front panel of the large plastic cover should fix the LCD display, button and switch to provide ease of access for the user. The gyroscope sensor should be attached parallel to the horizontal plane of the small plastic cover and fixed.

3 Devices and Elements Used

For the creation of the posture evaluation system, a multitude of devices and elements were connected to the MC of type Arduino Leonardo:

1. InvenSense 6-axis MPU-6050 gyroscope-accelerometer
2. LCD HD44780
3. Piezoelectric speaker
4. 9V battery
5. Electrical switch
6. External button
7. Large plastic case for device
8. Small plastic case for sensor

After wiring the elements together and connecting the necessary devices in I²C serial connection, the posture system was connected to a computer via USB-micro to USB cable. The Arduino open source software IDE, available for download on the official Arduino website, provided the basic environment for creating and uploading the functional code to the Arduino MC as well as monitoring and retrieving calibration values.

All of the elements of the posture system were then placed inside a protective plastic cover, providing the necessary means for fixation, portability and protection of the devices and connections inside.

The following chapters provide an overview of the used devices.

3.1 Arduino Leonardo



Figure 3.1. Arduino Leonardo microcontroller [9]

Arduino Leonardo is a microcontroller based on the ATmega32u4 datasheet. It is very similar to the more popular Arduino Uno, but differs from all preceding boards in that the ATmega32u4 has no need for a secondary processor with built in USB communication. This allows the Leonardo to appear to a connected computer as a mouse and keyboard, or it can also be seen as a virtual serial / COM port (Figure 3.1).

There are 20 digital input/output pins, of which 7 can be used as Pulse-width Modulation(PWM) outputs and 12 as analog inputs, a 16 MHz crystal oscillator, a micro USB connection, a power jack, an in-circuit serial programming (ICSP) header, and a reset button. Pins 2 and 3 are alternative pins for SDA and SCL respectively. SDA and SCL pins are present on the Arduino Leonardo as well. This allows for the connection of two devices without the need for an extension cable from a single pin.

Arduino Leonardo sources its power via USB cable, connected to a computer. In case both external power supply and USB are connected, the power source is selected automatically. [9]

Software is not required to be installed, as it is ready to use with it pre-installed inside of the package. The Arduino Leonardo is controlled by uploading code in the IDE and displaying information in the serial port window.

The full list of Arduino Leonardo technical specifications is presented below in Table 3.1.

Table 3.1 Arduino Leonardo Technical Specifications

Microcontroller	ATmega32u4
Operating voltage	5V
Input voltage	7-12V
Input voltage	6-20V
Digital I/O pins	20
Analog input channels	12
PWM channels	7
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB
Length	68.6 mm
Width	53.3 mm
Weight	20 g

Each of the 20 digital I/O pins on the Leonardo can be used as an input or output, using `pinMode()`, `digitalWrite()`, and `digitalRead()` functions. They operate at 5 volts and each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor of 20-50 kOhm. This value can be set for any desired input pin but the value is too large for the purpose of this project, therefore smaller pull-up resistors were connected to ground wires on some elements. [9]

3.2 InvenSense 6-axis MPU-6050 Gyroscope-Accelerometer

The first device connected to the MC is the InvenSense 6-axis MPU-6050 is a first of its kind motion tracking device that combines a 3-axis microelectromechanical system (MEMS) accelerometer, 3-axis MEMS gyroscope and a Digital Motion Processor (DMP) on a small 4x4x0.9mm board(Figure 3.2). It is very widely used for experiments related to precise motion measurements. The sensor records angle values rounded to the nearest hundredth of a degree based on built in gyroscope and gravity variable.

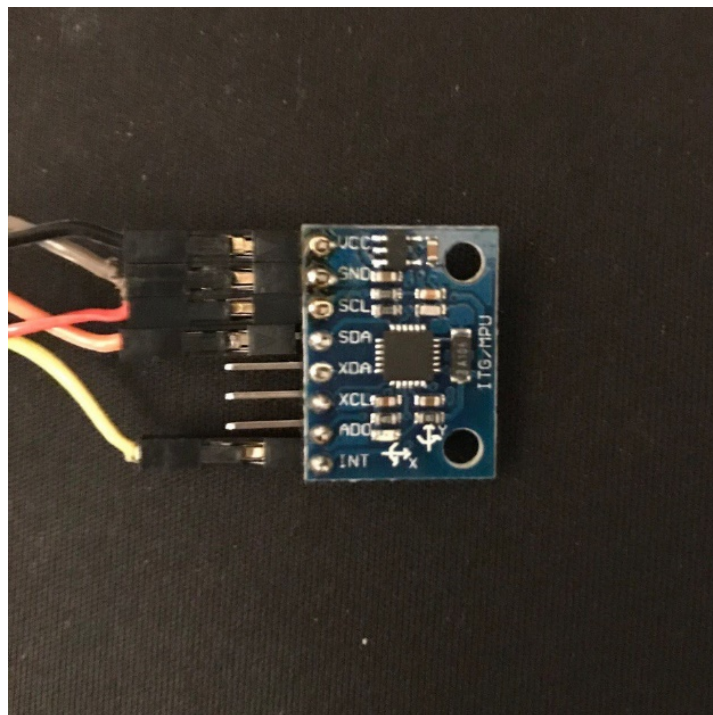


Figure 3.2. InvenSense 6-axis MPU-6050 gyroscope-accelerometer top view with connected wires

The MPU-6050 features three 16-bit analog-to-digital converters for digitizing the gyroscope outputs and three 16-bit analog to digital converters for digitizing the accelerometer outputs. For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ± 250 , ± 500 , ± 1000 , and $\pm 2000^\circ/\text{sec}$ and a user-programmable accelerometer full-scale range of $\pm 2g$, $\pm 4g$, $\pm 8g$, and $\pm 16g$. The standard gyroscope range of ± 250 was used for the purposes of this prototype. [10]

Table 3.2. Basic MPU-6050 characteristics (taken from [10])

Part/Item	MPU-6050
VDD	2.375V-3.46V
VLOGIC	1.71V to VDD
Serial Interfaces Supported	I ² C
Pin 8	VLOGIC
Pin 9	AD0
Pin 23	Bus-Clock(SCL)
Pin 24	Bus data line(SDA)

For this experiment, there was used only 1-axis “pitch” of the MEMS gyroscope element and the DMP.

The general characteristics of the 3-axis MEMS gyroscope important to the project are as follows:

- Digital-output X -, Y -,and Z -Axis gyroscopes with a user-programmable full scale
- Integrated 16-bit analog to digital enable simultaneous sampling of gyroscope
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Gyroscope operating current: 3.6mA

- Factory calibrated sensitivity scale factor

The general characteristics of the DMP important to the project are as follows:

- The MPU-6050 collects gyroscope and accelerometer data while synchronizing data sampling at a user defined rate. The total dataset obtained by the MPU-6050 includes 3-Axis gyroscope data, 3-axis accelerometer data, and temperature data. The MPU's calculated output to the system processor can also include heading data from a digital 3-axis third party magnetometer. The FIFO buffers the complete data set, reducing timing requirements on the system processor by allowing the processor to read the FIFO data. After burst reading the FIFO data, the system processor can save power by entering a low-power sleep mode while the MPU collects more data.
- Low-power pedometer functionality allows the host processor to sleep while the DMP maintains the step count. [4]

The MPU-6050 requires the following connection to perform the function of measuring the pitch and yaw angle relative to gravity. Interrupt pin set to 7(Appendix A).

Connection of the sensor is described in the following table:

Table 3.3. MPU-6050 Arduino pin connection

Arduino Leonardo Pin	MPU-6050
GND	GND
3V3	VCC
2	SDA
3	SCL
7	INT

The sensor was connected to the microcontroller according to the diagram depicted in Figure 3.3.

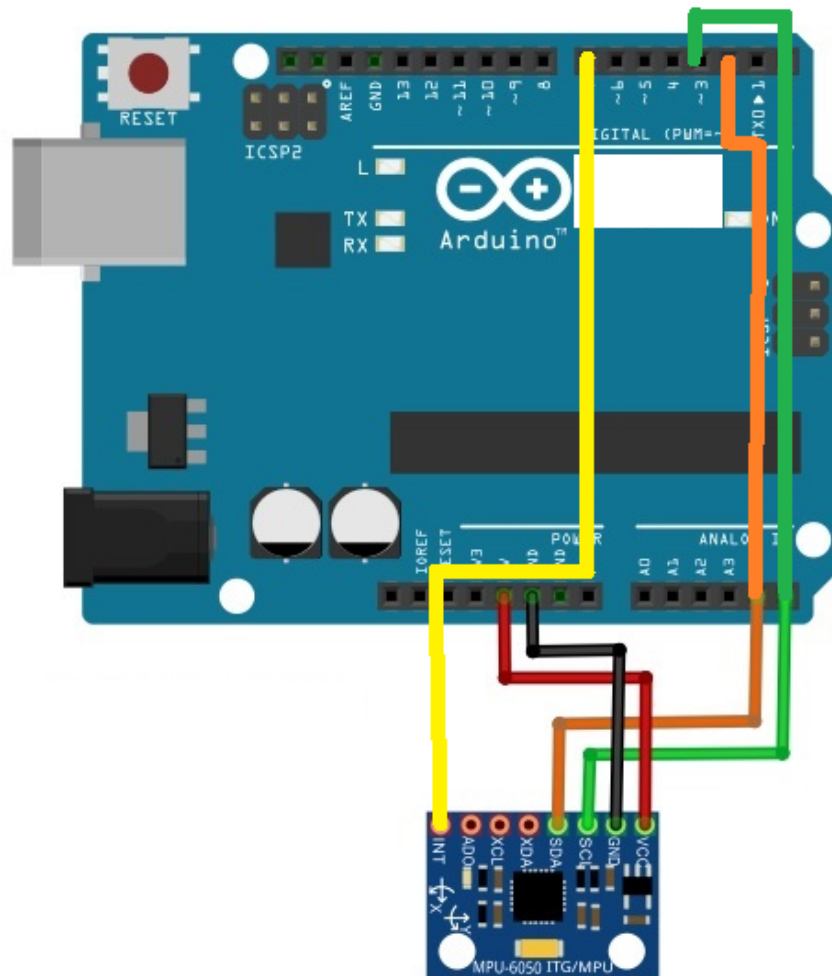


Figure 3.3. Connection used for MPU-6050(taken and modified from [9,10])

3.3 LCD HD44780

The second device to be connected to the MC is the 20x4 I²C LCD HD44780 display, one of the most common commercially available LCD displays used for simple display of text. The LCD front view with available characters with backlight turned on is shown in Figure 3.4.



Figure 3.4. LCD display front view

In Figure 3.5, the YWRobot module is shown connected to the 16 pins on the back of the LCD. It simplifies the connection from 16 pins to 4 and features a contrast adjustment knob and optional jumpers for backlight.

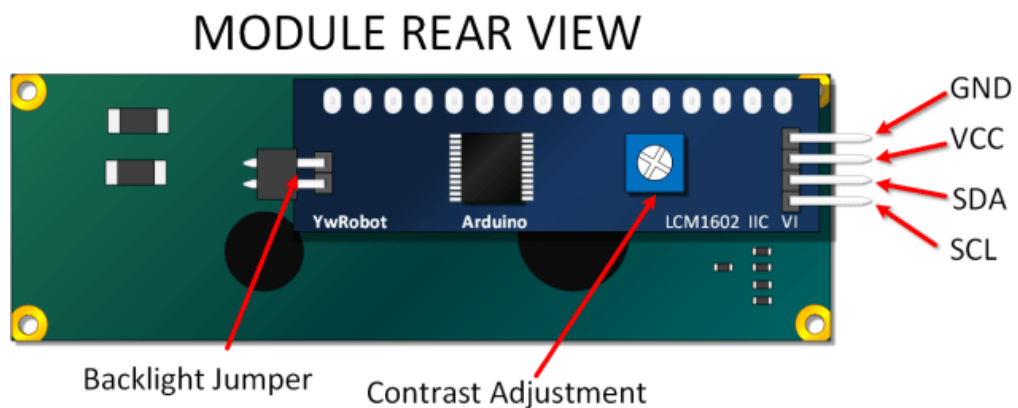


Figure 3.5. Rear view of the LCD HD44780 with YWRobot module connected to the 16 pins [11]

The 20x4 LCD HD44780 uses the 2-wire communication method of I²C and the device address is factory set to 0x27.

Connection of the LCD display to the microcontroller pins is described in Table 3.4.

Table 3.4 LCD Arduino pin connection

Arduino Leonardo Pin	20x4 I ² C LCD
GND	GND
+5V	VCC
2	SDA
3	SCL

The connection to the microcontroller is shown in Figure 3.5.

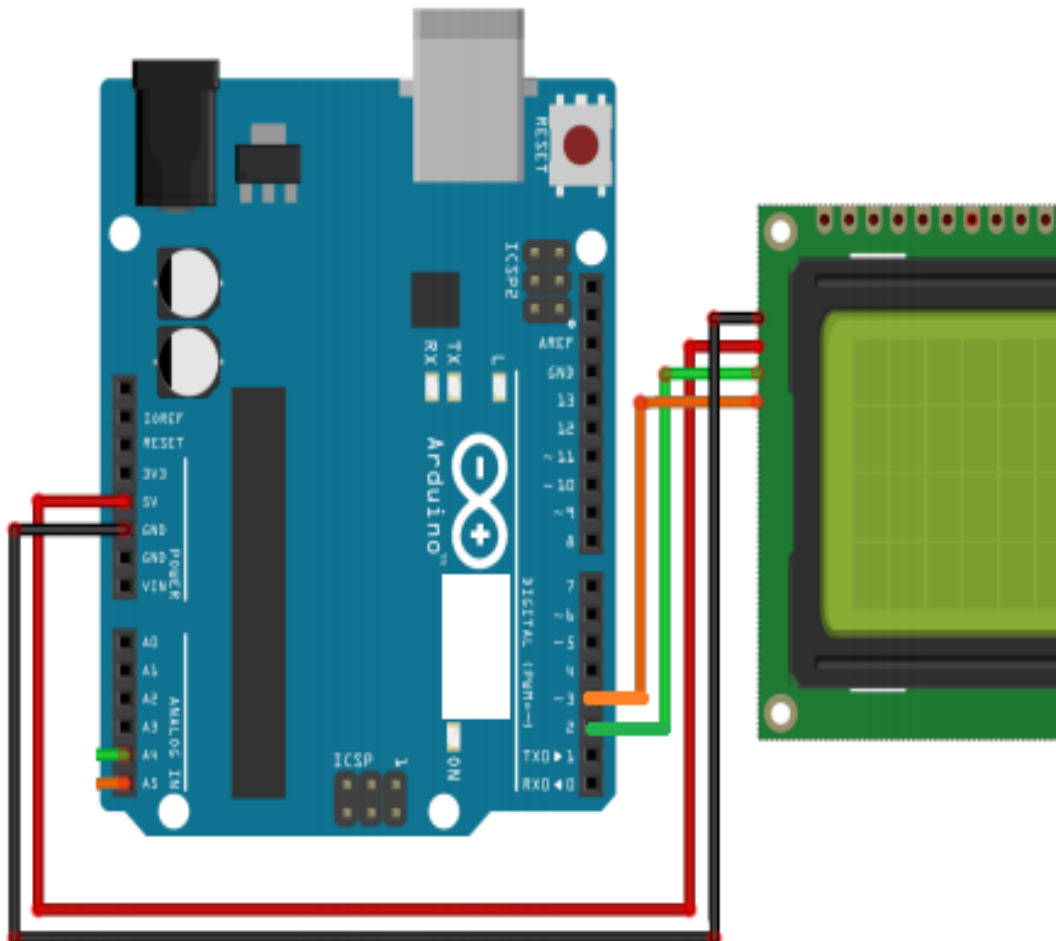


Figure 3.6. Connection used for 20x4 LCD display(taken and modified from [9])

3.4 Piezoelectric Speaker

The third component connected to the MC is a simple 8Ω piezoelectric speaker (Figure 3.7).



Figure 3.7. 8Ω piezoelectric speaker

Pin 8 was selected for communication with the speaker as “piezopin” and sound frequency set to 3kHz (see Appendix A). Low value of resistance over the speaker also requires a connection of a 100 Ohm pull-up resistor when connecting the other terminal to the ground wire. This eliminates the risk of causing a false alarm due to passage of a low current. The connection is described in Table 3.5 and depicted in Figure 3.8.

Table 3.5. Connection of speakers to microcontroller

Arduino Leonardo Pin	Piezoelectric speaker terminals
GND	Negative terminal of speaker connected through 100Ω pull-up resistor
8	Positive terminal of speaker

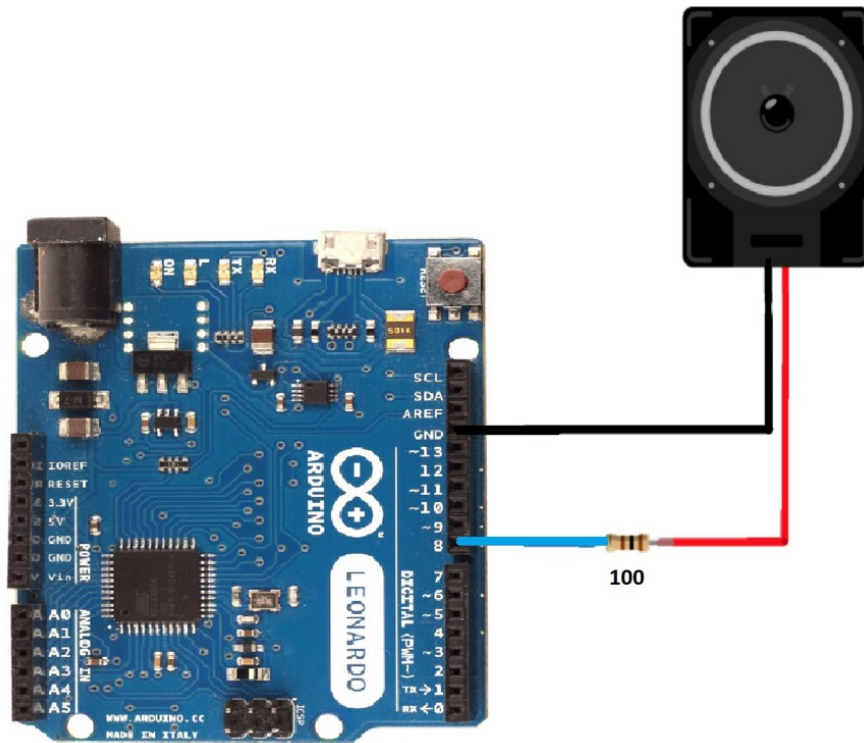


Figure 3.8. Speaker connection with Arduino with a 100 Ohm pull up resistor (taken and modified from [9])

3.5 9v Battery and Switch

The fourth component of the posture evaluation device prototype is the power source with a connected switch. The simple switch completes the circuit between the microcontroller and battery when powering the device “ON” and creates a short circuit when switched to cut off power supply, turning the device “OFF”. The connection of the 9V battery and switch with a snap connector is shown in Figure 3.9 and described in Table 3.6.

Table 3.6. Connection of battery and switch to the microcontroller

Arduino Leonardo Pin	Terminals
GND	Negative terminal of battery
Vin	Middle terminal of switch with positive terminal of battery connected on either side

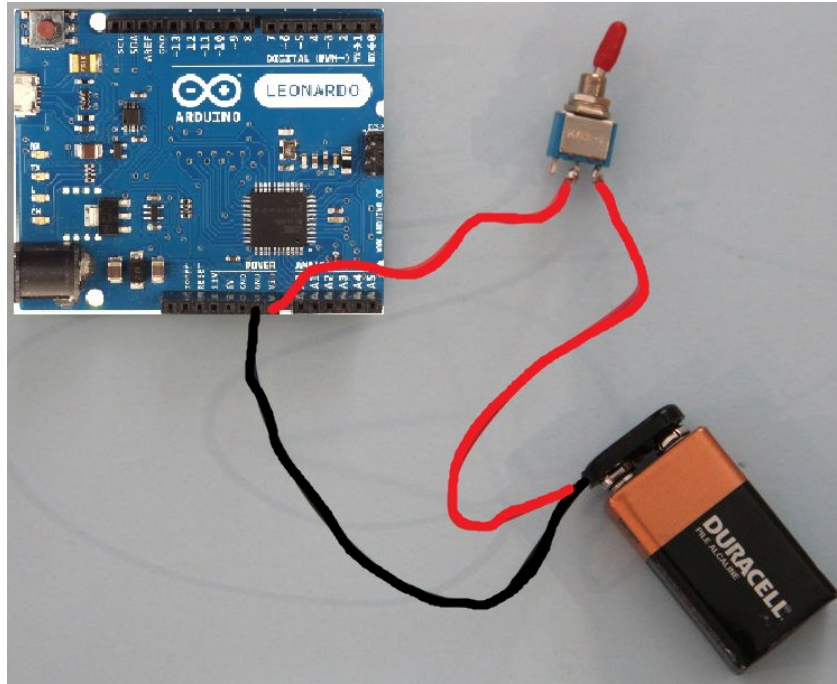


Figure 3.9. 9V Battery and switch connection to Arduino Leonardo with snap connector(taken and modified from[9])

3.6 External Button

The last component to be added to the device is the “sodial 100 pcs panel pcb momentary tactile tact push button switch 9963”. The connection of the external button is simply connecting one terminal to ground and the opposite terminal to the assigned button pin. The button pin was assigned to pin 5(see appendix A). When pressed, the button completes the circuit changing voltage state of the pin to LOW. It is included to initiate the posture evaluation process.

Connection is depicted in Figure 3.10.

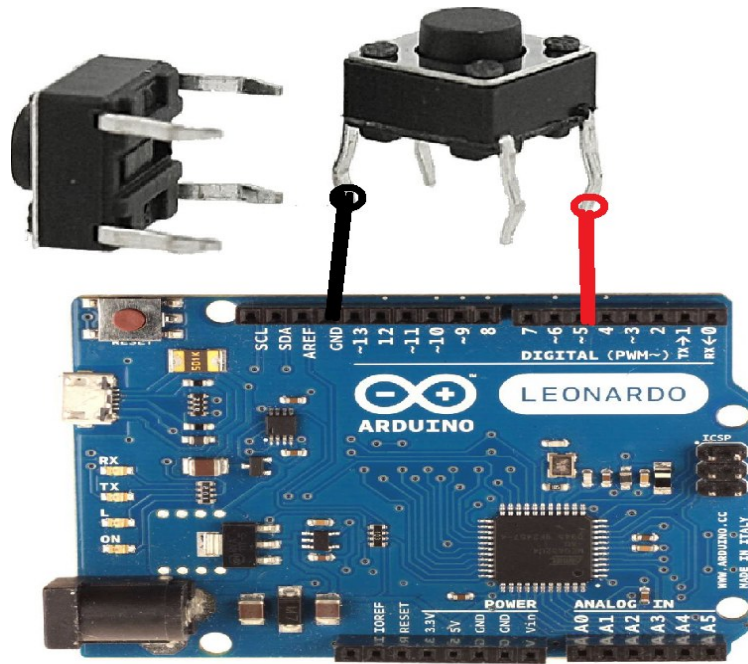


Figure 3.10. Connection of external button to Arduino Leonardo(taken and modified from [9])

3.7 Plastic Casing

In order to achieve optimal integrity of the device, a large black plastic case was used to enclose the main components of the device. A smaller black plastic case was used to enclose the sensor. The necessary cuts and holes in the large plastic casing were made with the help of an automatic power drill (Figures 3.11-12).



Figure 3.11. Plastic cover with drilled openings.

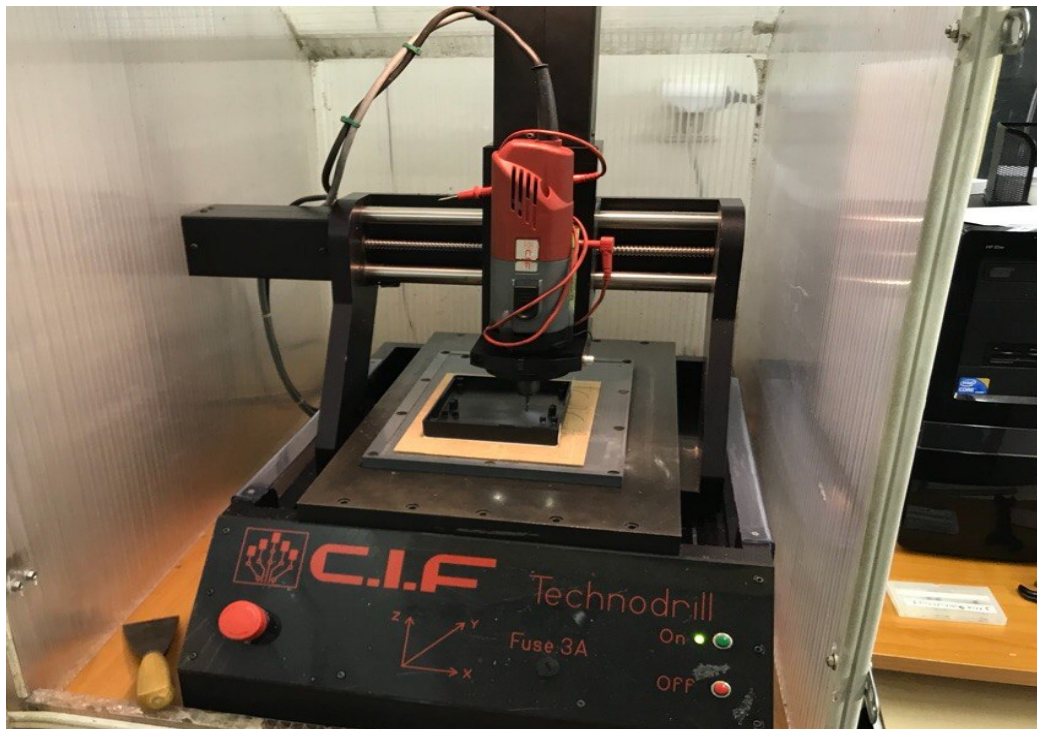


Figure 3.12 Automatic power drill with front panel placed on reference point

The front panel required areas to be cut out in order to fit the LCD, power switch and external button to the front panel and make them easily accessible for the user. Precise dimensions were drawn on the computer software connected to the drill and the cuts were delivered automatically. (Figure 3.13). All smaller adjustments, including the fitting of the sensor cable, were made by a flat file.

The most crucial part is that the software and drill will only give the proper result when the drawing was created according to a specific reference point and placed directly under it when drilling. Drilling frequency set to low value to eliminate melting of plastic yielding unrefined edges.

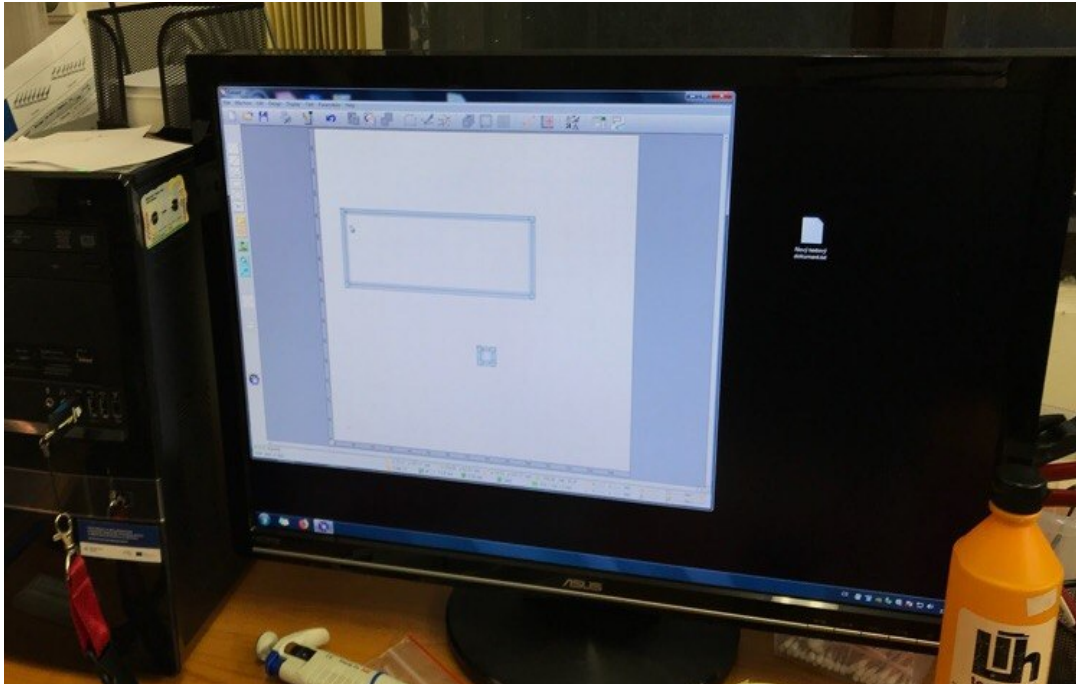


Figure 3.13. Automatic power drill software drawing of the front panel with cuts

All of the components of the prototype, with exception of the motion sensor, were then fixated inside the plastic cover with hot glue and double-sided tape. The LCD display, switch and external button were attached to the front panel with holes. The MC, 9V battery and Piezoelectric speaker were attached to the back panel. The microcontroller was fixated specifically to allow room for a USB connection to be possible, without compromising the integrity of the device.

The sensor was fixated directly onto the surface of the small plastic cover and sealed shut with hot glue. (Figure 2.13). Fixation of the cable was also provided by hot glue. The large plastic cover enclosing the device was then sealed shut with four screws on all corners from the bottom using a drill.

4 Results

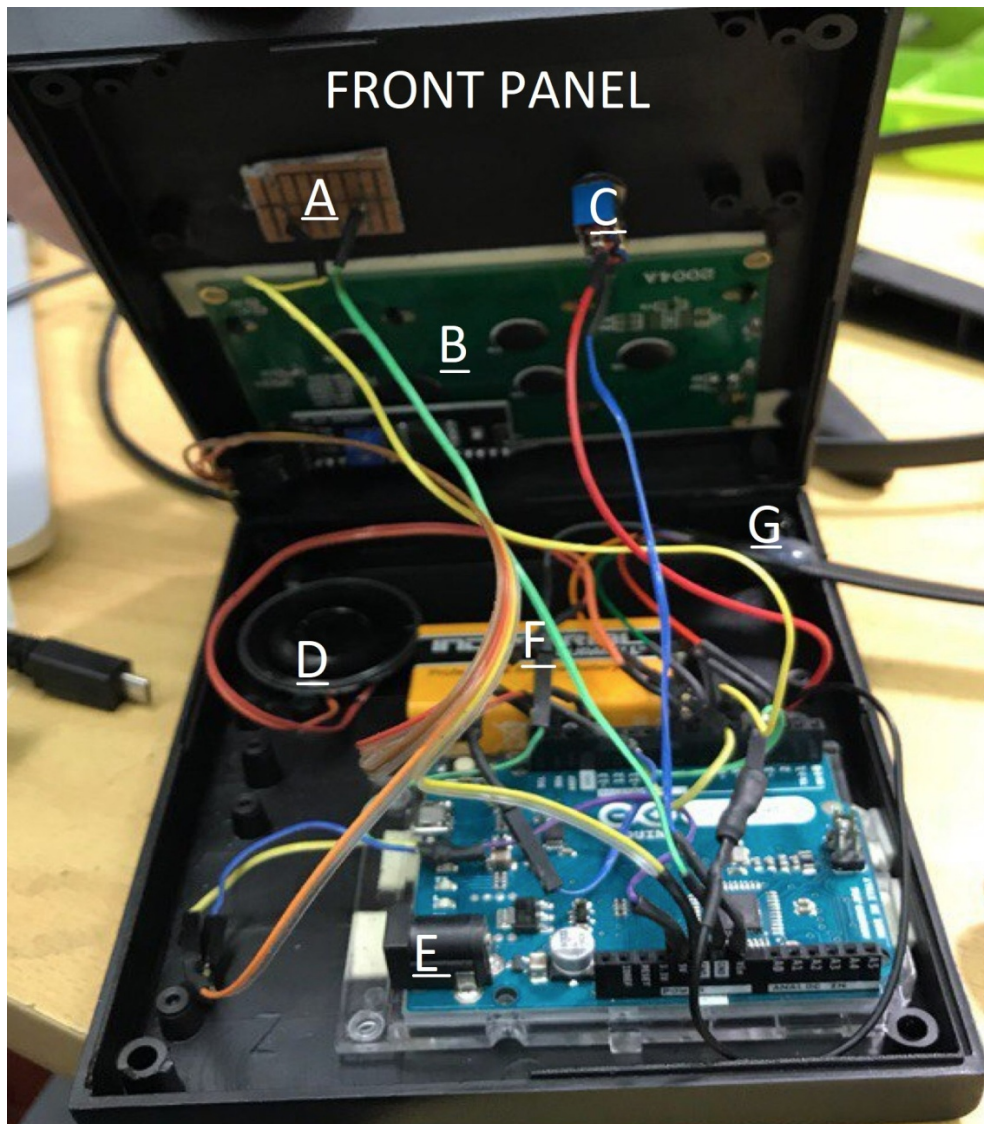


Figure 4.1 Posture evaluation device fixated inside plastic casing , where A— push button, B— LCD display, C—battery switch, D—piezoelectric speaker, E—Arduino Leonardo microcontroller, F—9V battery, G—sensor cable

Top view of the encased device with LCD, switch, MEMS sensor in plastic case and push button interfaced with the front panel are shown in Figures 4.2 and 4.3.



Figure 4.2. Posture evaluation device prototype

The device is powered on by flipping the red switch located on the front panel. Also located on the front panel next to the switch is the push button intended to initiate posture evaluation when pressed. The signal from the sensor is transmitted through a cable and processed by Arduino Leonardo microcontroller. The device creates an audible noise signal when values read from the sensor are above threshold levels set before use. This alarm serves as a cue for the user to improve posture. While user maintains proper posture, noise subsides and the LCD displays text indicating posture state and current deviation values.



Figure 4.3. Motion sensor in plastic cover

The sensor and connected cable are secured within the plastic cover, providing ease of manipulation and confidence in the duration of its' functionality. The plane of contact with human skin is opposite to where the cable and cap are fixed.

5 Calibration and Testing

Before any testing could take place, it is crucial to perform proper calibration of the MEMS gyroscope sensor in order to receive precise measurements. A software code was uploaded to the microcontroller mapping angle of inclination values from 0-180° in y and z (pitch and yaw) directions relative to gravity (see appendix A). Values from the calibrated sensor were then compared to real measured values. Testing on human subjects could begin only when the devices ability to measure angles of inclination precisely was verified. Angles of inclination were then converted to a percent value of deviation from the vertical axis in y and z directions by the modifying the software using formula:

$$(ydev, xdev) = \left| 100 * \left(1 - \frac{(y, z)}{90} \right) \right| \% \quad (5.1)$$

Healthy subjects were then selected to test the posture evaluation prototype and fill out a form by the end of the trial. A detailed description of the process of calibration, verification and finally testing on human subjects is covered in the following chapters.

5.1 Calibration and Verification

Calibration begins with running the I²C scanner code and initiating devices connected to the bus. This is done to verify functional connection of the SDA and SCL lines of the LCD display and sensor to the MC (see Appendix C).

An important note is that it is impossible to construct virtually identical 3-axis MEMS gyroscope sensors with such precision, therefore every sensor reads unique RAW values when tested. RAW values are read by the RAW code and displayed in the serial monitor of the Integrated Development Environment (IDE) for Arduino software (see Appendix B). Results shown on the serial monitor represent offsets of the 6 axes: x -axis accelerometer, y -axis accelerometer, z -axis accelerometer, x -axis gyroscope, y -axis gyroscope and z -axis gyroscope respectively (Figure 5.1).

COM3 (Arduino Leonardo)

a/g:	1288	36	15724	-771	-13	171
a/g:	1292	-16	15944	-750	-63	168
a/g:	1276	36	15860	-753	-51	172
a/g:	1272	76	15860	-785	-22	189
a/g:	1292	-8	15672	-768	-49	163
a/g:	1328	-84	15960	-754	-27	169
a/g:	1416	16	15812	-809	-57	170
a/g:	1340	-16	15820	-758	-42	163
a/g:	1296	64	15768	-752	-6	188
a/g:	1344	4	15864	-752	-54	169
a/g:	1348	-24	15684	-810	-46	162
a/g:	1292	12	15888	-756	-28	172
a/g:	1316	-24	15916	-765	-29	158
a/g:	1372	28	15764	-761	-27	178
a/g:	1392	4	15788	-775	-38	175
a/g:	1276	-8	15932	-765	-14	151
a/g:	1268	4	15960	-759	-22	170
a/g:	1332	-12	15752	-778	-68	186
a/g:	1244	20	15808	-767	-19	162
a/g:	1356	16	15752	-778	-39	172
a/g:	1264	-52	15780	-764	-36	175
a/g:	1420	108	15800	-763	-24	179
a/g:	1360	88	15748	-755	-49	171
a/g:	1328	44	15912	-762	-46	174
a/g:	1328	-44	15732	-769	-49	169
a/g:	1320	0	15808	-772	-38	174

Figure 5.1. Offset values obtained from “Raw” code in MPU6050 library

Once it is verified that the sensor is functional, calibration of digital motion processor offsets may be performed. Prior to running the DMP offsets code (see Appendix D), the gyroscope motion sensor must be placed in a horizontal position relative to the direction of gravity (Figure 5.2).

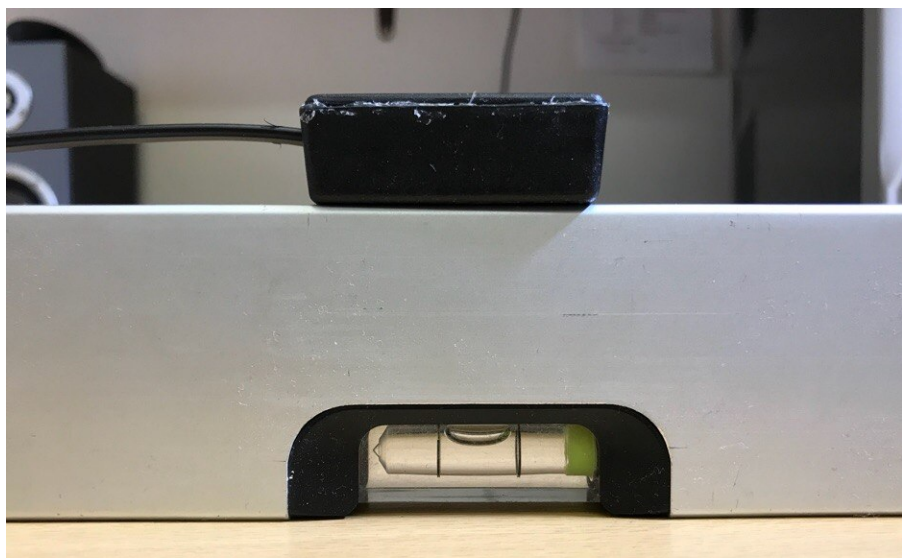


Figure 5.2. Sensor placed on level parallel to horizontal plane relative to gravity

After running the DMP offsets code, the offset values for all 6 axes of the sensor are received (Figure 5.3)

```
Sensor readings with offsets:  -1      8      16381  0      2      0
Your offsets:                 -4742  -3936  1555   192    9     -48
```

Figure 5.3. DMP offset results in serial monitor

These offsets were then edited into the main posture evaluation code with the `mpu.setoffset` functions (see Appendix A). In order to verify the precision of the calibration, a test was performed to compare measured and real values read by the motion sensor with a precision error to the nearest degree. Verification of calibration test depicted in Figure 5.4. The protractor tool was set directly parallel to the horizontal plane mediated by the level tool. The test consisted of mounting the sensor onto the angle measuring tool and verifying relative angle measurements of yaw direction angles of 0, 45, 90, 135 and 180 degrees relative to the horizontal plane. The same process was repeated for the angle in y direction for pitch, until the minimum error of 0 was achieved. All results from verification of calibration are depicted in Table 5.1.



Figure 5.4. Verification of calibration of $z(\text{yaw})$ angle using a protractor and level

5.2 Human Subject Testing

The main software code responsible for evaluation of deviation angles (see Appendix A) was then uploaded and the device was programmed to display accurate postural deviation angles and indicators describing suggested corrections (Figure 5.5).

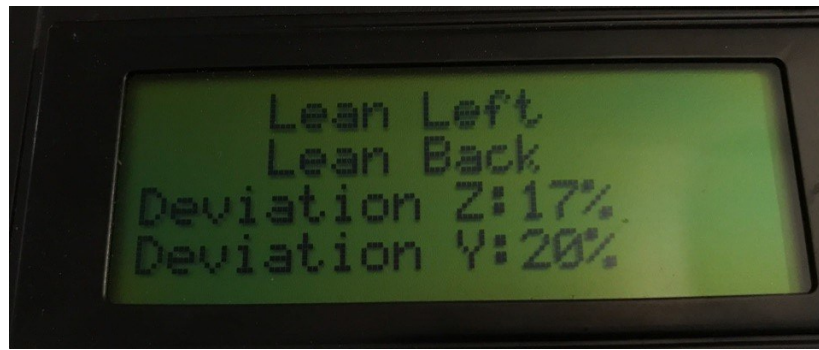


Figure 5.5. LCD display of posture evaluation with percent of deviation values with lean indicators

14 volunteers were selected to take part in testing the posture evaluation prototype. The subjects were required to use the device and fill out a form of questions (see Appendix E). The test process can be described in 2 steps:

1. Posture Evaluation Session
2. Questionnaire/Form

5.2.1 Posture Evaluation Session

The posture evaluation session required 14 participants to use the posture evaluation device for a minimum of 20 minutes while performing daily tasks involving sitting and standing for extended periods of time. Tasks performed by participants included walking, working with computer, using a mobile phone, brushing teeth and in some cases cooking. The individuals were required to monitor the noise signal and deviation values shown on the LCD. Adjustments were to be made by the subject based on these cues.



Figure 5.6. Human test participant engaging in daily activities with mounted sensor

The position of the sensor was selected based on current existing devices of similar type. All participants were required to place the sensor on the upper back between the shoulder blades, where the curve of the spine is nearest to vertical alignment. Fixation of the sensor was performed by placing double-sided tape between the sensor and the surface of the skin (Figure 5.7).



Figure 5.7. Positioning of sensor on upper back between the shoulder blades

5.2.2 Questionnaire/Form

Directly after trialing the device, the subjects were asked to fill out a questionnaire. The appropriate answer to each question is a subjective score ranging from 1-5. Filling out an answer of 1 translates to “strongly disagree”, 2 is “disagree”, 3 is “neutral”, 4 is “agree”, rising up to 5, which translates to “strongly agree”. All answers were averaged and the results input into Table 6.2.

6 Testing Results

Table 6.1 Verification of calibration to nearest degree

System Measurement (°)	Protractor measurement (°)	Error(Δ)
0	0	0
45	45	0
90	90	0
135	135	0
180	180	0

Table 6.2 Human subject testing questionnaire results

Question number	Averaged answer score
1	5.00
2	5.00
3	4.64
4	3.14
5	4.50
6	4.57
7	5.00
8	3.86
9	4.21
10	4.64

7 Discussion

The designed device proved successful in performing the required functions, although there were many problems encountered throughout the process. One of the main issues encountered was disconnection LCD and sensor from the I²C bus. This was found to be caused by absence of physical integrity of the device and its connections. Sensors were easily damaged and corrupted when manipulating them. A total of 4 MPU-6050 gyroscope-accelerometers were purchased for this device.

The experiment required for the LCD display and 6-axis accelerometer SDA and SCL ports to be connected to the Arduino Leonardo MC with soldered wires and cables to the digital outputs 2, 3. Leonardo uses these pins for SDA and SCL, while the most common micro-controllers usually use analog inputs 4 and 5 respectively for these inputs.

It was evident that before making any measurements with human subjects it was necessary to protect and fixate the device and sensor in a plastic cover. Cover dimensions were chosen after creating an abstract measurement of the devices intended to fit inside. Dimensions of the device were mainly dictated by the size of the LCD display.

After covering the device, the sound signal became barely noticeable. Adjustments had to be made to the frequency of the audio signal, as there are no openings in the plastic cover for the sound to pass through.

Fixating the sensor also presented a challenge. It was crucial for the plane of the sensor and the plane of the plastic cover to be precisely parallel and nearly coplanar. Placing and fixating the cable connecting the sensor to the microcontroller required careful consideration of device dimensions and possible physical damage.

Supplying power to the Arduino Leonardo via USB cable was not a viable option, as it would impair the prototypes' portability and make it dependent on a connection to some power source with a USB port. Therefore, a battery with a snap connector connected to the Arduino Leonardo MC through a switch was selected to perform the role of controlled external power source. The assumption is that the battery would be cheap and easily fixated inside the device. The snap connector provides the possibility to replace the battery upon depletion.

The values read by the sensor were converted from radians to degrees in range 0-180 by the applied software code. It was then modified to show a percent value of deviation from the vertical axis, which is more applicable for user subjective evaluation of posture (see Appendix A).

When running the RAW code for the MPU-6050 the values read are chaotic and must take many variables in account. Therefore, it was necessary to wait approximately 10 seconds for values shown on the serial monitor to balance out. The values for each individual sensor are factory made unique and offsets for all 6 axes must be accounted for prior to performing any measurements.

The calibration process of the posture evaluation prototype proved to be successful as we can see from the compared values and errors in Table 6.1. Values measured by the sensor directly reflected the angle values shown on the protractor tool. Although we could only verify the measurement with a precision value to the nearest degree, this did not impair the overall functionality of the device. If we take into consideration the possibility of human error when aligning the centers of the axes and other measurement uncertainties, we remain confident in the fact that the sensor is extremely precise. It was evident from the beginning of the test and anecdotal evidence, that the sensor was extremely accurate in calculating angle of inclination. This model is very often used to control quadcopters flight, which requires extremely precise measurements of the pitch, roll and yaw to adjust during flight.

The use of the Leonardo model of the Arduino MCs had some advantages and disadvantages. There was a power adapter included in the packaging from the beginning, but the presence of the ATmega32u4 serial ports and native USB greatly simplified the construction of the prototype. The USB communication on ATmega32u4 is integrated on the board making it cheaper and more efficient, as opposed to Arduino Uno, which must translate the USB signal into a serial communication that the native ATmega328 could understand. The device was connected and programmed through built-in micro USB communication before any battery was implemented. Unfortunately, the mass majority of information regarding the Arduino MC is centered on the Arduino Uno, making it difficult to find relevant information for Leonardo since there are differences in analog and digital inputs discussed in Chapter 2, making the connection process slightly perplexing.

Testing of human subject presented some challenges. Initially five healthy participants were selected, but during the process nine more healthy individuals decided to partake in the experiment. This was beneficial for gathering more data regarding the subjective functionality of the device, but required devoting more time for testing and guiding the human subjects through the process.

Appropriate results and signal of the device was heavily influenced by the position of the sensor. With advisory from the consultant of the thesis MUDr. Markéta Janatová, sensor placement position was selected between the shoulder blades in a vertical position. In some cases, the sensor was initially fixed in a wrong position and required readjustment. In other cases, the sensor was detached by pulling the cord with movement of arms and from catching on surrounding objects. This posed some obvious problems and required the testing process to be reinitiated.

The results from first 5 questions on the questionnaire were the most significant. A score of 5.00 on the first two questions of this project shows that this device definitely motivates the user to adjust posture. This also shows that the dynamic deviation values allowed the user to obtain more information on his/her position in space.

Questions number 3 and 4 ask whether the user experienced positive posture formation during and after the posture evaluation session respectively. Positive effect during testing received a high score of 4.64, meaning most participants found the device beneficial for maintaining posture while using the device. Positive effect after using of the device received an average score of 3.14, signifying that after using the device for 20 minutes postural habits were unaffected. This is most likely due to the duration of the posture evaluation session being 20 minutes in length.

The next questions 5 and 6 asked the user to evaluate the alarm system. A score of 4.50 and 4.57 shows that users agree that the alarm system was functional and accurate for the most part.

Question 7 and 8 cover the accessibility and comfort levels while evaluating posture, receiving scores of 5.00 and 3.14 respectively. All participants found that the interface of the device was easy to use, while only half agreed that the sensor was easy to wear. This can be attributed to obvious reasons such as presence of a cable and requirement to use double-sided tape for fixation.

The last questions required the participants to give a subjective view on the prolonged usage of the device and the likeliness of using it in the future. The last questions received a score of 4.21 and 4.64 respectively, meaning most participants

found the device potentially more beneficial with longer use. Most participants agreed they would be likely to use the final version of this product in prospect to promote healthy postural habit formation.

As was stated in the Chapter 1, habit formation is constituted by repeating an action consistently. As it was very difficult to perform consistent tests for longer periods of time on the same subjects, very little evidence is seen on changing postural habits after using the device for such a short duration. However, continuous use of this device for a duration at least 3 weeks is expected to promote postural habit formation. [7]

8 Conclusion

The performed experiments allow to conclude that this portable device detects and displays precise posture misalignment, is likely to promote positive postural habit formation and does not require a connection to a mobile application. The posture system designed in this project can be used to evaluate the angle of inclination and fault of posture of any tilting object, as long as it lays coplanar to the vertical plane of alignment. By connecting the sensor to the object and processing information in the MC, the system determines whether the connected object has proper or improper posture, sending all information to the LCD display. Threshold values can be adjusted to fit the needs of the subject. The tests performed prove that this setup can easily be applied to measure a precise angle of inclination for various purposes and detecting vertical angular deviation in z and y directions within any system that that the sensor is properly attached to, including the human body.

During all the processes involving designing and connecting of the posture evaluation device there were many notes taken to improve on it. First, the device could be constructed to be much smaller if the LCD used would have smaller dimensions. There is no imperative need for such a relatively large LCD display, speaker and battery in the context of this project. Second, the signal of posture fault could be extended to include a vibrating element inside the plastic cover containing the sensor. A vibrating cue seems to be more compatible with human interactions and is more intuitive regarding position of the signal. Finally, the device could be extended to include connection to a smartphone via Bluetooth Adapter. This would not impair the functionality of the prototype independent of connection to the phone. The phone would utilize a simple mobile application to interface with the posture evaluation device. The signal of posture fault could then manifest as a vibration of the users' smartphone in their pocket.

References

- [1] Clark M, Lucett S, Sutton BG. *NASM Essentials of Personal Fitness Training* 4th edition revised. Philadelphia: Wolters Kluwer Health/Lippincott Williams & Wilkins; 2014.
- [2] *International Journal of Physical Education, Sports and Health* [online] 2016; 3(1):177-178 ;Available at: <https://www.ncbi.nlm.nih.gov/pubmed/11926755>[Accessed 26.01.2018.]
- [3] Carter JB, Banister EW: Musculoskeletal problems in VDT work: a review. *Ergonomics*, 1994, 37: 1623–1648.
- [4] Curnow D, Cobbin D, Wyndham J: Altered motor control, posture and the Pilates method of exercise prescription. *J Body MovTher*, 2009, 13: 104–111.
- [5] Kim JK, Lee SJ:Effect of stretching exercise as work-related musculoskeletal pain of neck and shoulder. *Korean J Phys Edu*, 43: 655–662.
- [6] Neal DT, Wood W, Labrecque JS, Lally P. How do habits guide behavior? Perceived and actual triggers of habits in daily life. *J ExpSoc Psychol*. 2012;48:492–498.
- [7] Lally P, Gardner B. Promoting habit formation. *Health Psychology Rev*. In press: DOI: 10.1080/17437199.2011.603640.
- [8]Apparatus to serve as a reminder for posture improvement US 20050070830 A1.[online]2018. Available at: <https://www.google.com/patents/US20050070830>. [Accessed 26.01.2018].
- [9] ARDUINO LEONARDO WITH HEADERS [online] 2018 Available at: <https://store.arduino.cc/arduino-leonardo-with-headers> [Accessed 26.01.2018]

[10] InvenSense Inc., MPU-6000/MPU-6050 Product Specification;PS-MPU-6000A-00,[online] 19.8.2013 Available at: <https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf> [Accessed 26.01.2018]

[11] YWRobot LCM1602 IIC V1 LCD Arduino Tutorial[online] 2018 Available at: <http://henrysbench.capnfatz.com> [Accessed 5.05.2018]

Appendix A. Main code for posture evaluation

```
#include<Wire.h>
#include <LiquidCrystal_I2C.h>
#include <I2Cdev.h>
constintMPU_addr=0x68; int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;

intminVal=265;
intmaxVal=402;
int boot=0;
intbuttonPin=5;
intpiezoPin = 8;
double x; double y; double z;

#include "MPU6050_6Axis_MotionApps20.h"
MPU6050 mpu;

#define I2C_ADDR 0x27
#define BACKLIGHT_PIN 3
#define En_pin 2
#define Rw_pin 1
#define Rs_pin 0
#define D4_pin 4
#define D5_pin 5
#define D6_pin 6
#define D7_pin 7

LiquidCrystal_I2C lcd(I2C_ADDR, En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin,
POSITIVE);

#define MPU6050_INT_PIN 7
#define MPU6050_INT digitalPinToInterrupt(MPU6050_INT_PIN)

volatile boolmpuInterrupt = false;
voidddmpDataReady() {
mpuInterrupt = true;
}
#define MPU6050_INT_PIN 7
#define MPU6050_INT digitalPinToInterrupt(MPU6050_INT_PIN)

voidsetup()
{
lcd.begin(20,4);
lcd.setCursor ( 0, 0 );
lcd.print("Posture Evaluation");
lcd.setCursor ( 0, 1 );
lcd.print("BachelorThesis");
lcd.setCursor (0,2);
```



```

lcd.print("By: CornilPetrov");
lcd.setCursor(0,3);
lcd.print("Ing. Tomas Funda,PhD");
delay(2000);

Serial.begin(14400);
Wire.begin();
Wire.beginTransmission(MPU_addr);
Wire.write(0x6B);
Wire.write(0);
Wire.endTransmission(true);

pinMode(buttonPin, INPUT_PULLUP);

mpu.setXGyroOffset(51);
mpu.setYGyroOffset(83);
mpu.setZGyroOffset(211);
mpu.setXAccelOffset(-4332);
mpu.setYAccelOffset(1731);
mpu.setZAccelOffset(474);
}

void loop() {

if(digitalRead(buttonPin) == LOW)
{
if(boot == 0)
{
boot = 255;
lcd.clear();
}
else
{
boot = 0;

}
}
}
if(boot > 0)
{
Wire.beginTransmission(MPU_addr);
Wire.write(0x3B);
Wire.endTransmission(false);
Wire.requestFrom(MPU_addr,14,true);
AcX=Wire.read()<<8|Wire.read();
AcY=Wire.read()<<8|Wire.read();
AcZ=Wire.read()<<8|Wire.read();
intxAng = map(AcX,minVal,maxVal,-90,90);
intyAng = map(AcY,minVal,maxVal,-90,90);
intzAng = map(AcZ,minVal,maxVal,-90,90);

```

```

int y= (1-((RAD_TO_DEG * (atan2(-xAng, -zAng)+PI)-180)/90))*100;
int z= (1-((RAD_TO_DEG * (atan2(-xAng, -yAng)+PI)-180)/90))*100;
lcd.setCursor(0,0);
lcd.print(" ");
lcd.setCursor(0,3);
lcd.print(" ");
lcd.setCursor ( 0, 2 );
lcd.print("Deviation Z:");
lcd.print(abs(z));
lcd.print("% ");
lcd.setCursor ( 0, 3 );
lcd.print("Deviation Y:");
lcd.print(abs(y));
lcd.print("% ");
delay(250);
lcd.setCursor ( 12, 2 );
lcd.print(" ");
lcd.setCursor ( 12, 3 );
lcd.print(" ");

```

```

if(y>10)
{
lcd.setCursor(0,1);
lcd.print(" Lean Back ");
}
else if(y<-10)
{

lcd.setCursor(0,1);
lcd.print(" Lean Forward ");

}
if(z>10)
{
lcd.setCursor(0,0);
lcd.print(" Lean Right ");

}
else if (z<-10)
{
lcd.setCursor(0,0);
lcd.print(" Lean Left ");

}
if (abs(y)<10 && abs(z)<10)
{
lcd.setCursor(0,0);
lcd.print(" Great ");
}

```

```

lcd.setCursor(0,1);
lcd.print("  Posture  ");
tone(piezoPin, 0, 250);
delay(250);
lcd.setCursor(0,0);
lcd.print("          ");
lcd.setCursor(0,1);
lcd.print("          ");
}
if (abs(y)>10)
{
tone(piezoPin, 3000, 250);
}
else if(abs(z)>10)
{
tone(piezoPin, 3000, 250);
}
if (abs(y)<10)
{
lcd.setCursor(0,1);
lcd.print("          ");
}
if (abs(z)<10)
{
lcd.setCursor(0,0);
lcd.print("          ");
}
delay(500);
}
}

```

Appendix B Code for retrieving RAW values

```
#include "I2Cdev.h"
#include "MPU6050.h"

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE
// implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for InvenSense evaluation board)
// AD0 high = 0x69
MPU6050 accelgyro;
//MPU6050 accelgyro(0x69); // <— use for AD0 high

int16_t ax, ay, az;
int16_t gx, gy, gz;

// uncomment "OUTPUT_READABLE_ACCELGYRO" if you want to see a tab-
// separated
// list of the accel X/Y/Z and then gyro X/Y/Z values in decimal. Easy to read,
// not so easy to parse, and slow(er) over UART.
#define OUTPUT_READABLE_ACCELGYRO

// uncomment "OUTPUT_BINARY_ACCELGYRO" to send all 6 axes of data as
// 16-bit
// binary, one right after the other. This is very fast (as fast as possible
// without compression or data loss), and easy to parse, but impossible to read
// for a human.
// #define OUTPUT_BINARY_ACCELGYRO

#define LED_PIN 13
bool blinkState = false;

void setup() {
// join I2C bus (I2Cdev library doesn't do this automatically)
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
Wire.begin();
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
Fastwire::setup(400, true);
#endif

// initialize serial communication
// (38400 chosen because it works as well at 8MHz as it does at 16MHz, but
```

```

// it's really up to you depending on your project)
Serial.begin(38400);

// initialize device
Serial.println("Initializing I2C devices...");
accelgyro.initialize();

// verify connection
Serial.println("Testing device connections...");
Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful":
"MPU6050 connection failed");

// use the code below to change accel/gyro offset values

Serial.println("Updating internal sensor offsets...");
// -76 -2359 1688 0 0 0
Serial.print(accelgyro.getXAccelOffset()); Serial.print("\t"); // -76
Serial.print(accelgyro.getYAccelOffset()); Serial.print("\t"); // -2359
Serial.print(accelgyro.getZAccelOffset()); Serial.print("\t"); // 1688
Serial.print(accelgyro.getXGyroOffset()); Serial.print("\t"); // 0
Serial.print(accelgyro.getYGyroOffset()); Serial.print("\t"); // 0
Serial.print(accelgyro.getZGyroOffset()); Serial.print("\t"); // 0
Serial.print("\n");
accelgyro.setXGyroOffset(0);
accelgyro.setYGyroOffset(0);
accelgyro.setZGyroOffset(0);
Serial.print(accelgyro.getXAccelOffset()); Serial.print("\t"); // -76
Serial.print(accelgyro.getYAccelOffset()); Serial.print("\t"); // -2359
Serial.print(accelgyro.getZAccelOffset()); Serial.print("\t"); // 1688
Serial.print(accelgyro.getXGyroOffset()); Serial.print("\t"); // 0
Serial.print(accelgyro.getYGyroOffset()); Serial.print("\t"); // 0
Serial.print(accelgyro.getZGyroOffset()); Serial.print("\t"); // 0
Serial.print("\n");

// configure Arduino LED pin for output
pinMode(LED_PIN, OUTPUT);
}

void loop() {
// read raw accel/gyro measurements from device
accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

// these methods (and a few others) are also available
//accelgyro.getAcceleration(&ax, &ay, &az);
//accelgyro.getRotation(&gx, &gy, &gz);

#ifdef OUTPUT_READABLE_ACCELGYRO
// display tab-separated accel/gyro x/y/z values

```

```

Serial.print("a/g:\t");
Serial.print(ax);Serial.print("\t");
Serial.print(ay); Serial.print("\t");
Serial.print(az); Serial.print("\t");
Serial.print(gx);Serial.print("\t");
Serial.print(gy);Serial.print("\t");
Serial.println(gz);
#endif

#ifdef OUTPUT_BINARY_ACCELGYRO
Serial.write((uint8_t)(ax » 8)); Serial.write((uint8_t)(ax& 0xFF));
Serial.write((uint8_t)(ay » 8)); Serial.write((uint8_t)(ay & 0xFF));
Serial.write((uint8_t)(az » 8)); Serial.write((uint8_t)(az& 0xFF));
Serial.write((uint8_t)(gx » 8)); Serial.write((uint8_t)(gx& 0xFF));
Serial.write((uint8_t)(gy » 8)); Serial.write((uint8_t)(gy & 0xFF));
Serial.write((uint8_t)(gz » 8)); Serial.write((uint8_t)(gz& 0xFF));
#endif

// blink LED to indicate activity
blinkState= !blinkState;
digitalWrite(LED_PIN, blinkState);
}

```

Appendix C I²C scanner

```
#include <Wire.h>

void setup()
{
  Wire.begin();

  Serial.begin(9600);
  while (!Serial); // Leonardo: wait for serial monitor
  Serial.println("\nI2C Scanner");
}

void loop()
{
  byte error, address;
  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {
    Wire.beginTransmission(address);
    error = Wire.endTransmission();
    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println(" !");

      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Unknown error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(5000);}
}
```

Appendix D code for processing DMP offsets

```
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

////////////////////// CONFIGURATION ////////////////////////
intbufferSize=1000; //Amount of readings used to average, make it higher to get
more precision but sketch will be slower (default:1000)
intaccel_deadzone=8; //Acelerometer error allowed, make it lower to get more
precision, but sketch may not converge (default:8)
intgiro_deadzone=1; //Giro error allowed, make it lower to get more precision, but
sketch may not converge (default:1)

MPU6050 accelgyro(0x68); // 100 &&i<=(bufferSize+100)} //First 100 measures
are discarded
buff_ax=buff_ax+ax;
buff_ay=buff_ay+ay;
buff_az=buff_az+az;
buff_gx=buff_gx+gx;
buff_gy=buff_gy+gy;
buff_gz=buff_gz+gz;
}
if (i==(bufferSize+100)){
mean_ax=buff_ax/bufferSize;
mean_ay=buff_ay/bufferSize;
mean_az=buff_az/bufferSize;
mean_gx=buff_gx/bufferSize;
mean_gy=buff_gy/bufferSize;
mean_gz=buff_gz/bufferSize;
}
i++;
delay(2); //Needed so we don't get repeated measures
}
}

void calibration(){
ax_offset=-mean_ax/8;
ay_offset=-mean_ay/8;
az_offset=(16384-mean_az)/8;

gx_offset=-mean_gx/4;
gy_offset=-mean_gy/4;
gz_offset=-mean_gz/4;
while (1){
int ready=0;
accelgyro.setXAccelOffset(ax_offset);
accelgyro.setYAccelOffset(ay_offset);
accelgyro.setZAccelOffset(az_offset);
```



```

accelgyro.setXGyroOffset(gx_offset);
accelgyro.setYGyroOffset(gy_offset);
accelgyro.setZGyroOffset(gz_offset);

meansensors();
Serial.println("...");

if (abs(mean_ax)<=accel_deadzone) ready++;
else ax_offset=ax_offset-mean_ax/accel_deadzone;

if (abs(mean_ay)<=accel_deadzone) ready++;
else ay_offset=ay_offset-mean_ay/accel_deadzone;

if (abs(16384-mean_az)<=accel_deadzone) ready++;
else az_offset=az_offset+(16384-mean_az)/accel_deadzone;

if (abs(mean_gx)<=giro_deadzone) ready++;
else gx_offset=gx_offset-mean_gx/(giro_deadzone+1);

if (abs(mean_gy)<=giro_deadzone) ready++;
else gy_offset=gy_offset-mean_gy/(giro_deadzone+1);

if (abs(mean_gz)<=giro_deadzone) ready++;
else gz_offset=gz_offset-mean_gz/(giro_deadzone+1);

if (ready==6) break;
}
}

```

Appendix E Testing Questionnaire

1. This device reminded me to adjust my posture.
2. The displayed deviation values helped provide more detailed information about my current position.
3. This device affected my posture in a positive way while using it.
4. This device affected my posture in a positive way after using it.
5. This device alarmed me every time posture fault was present.
6. This device did not alarm me when posture fault was not present.
7. The device interface was easy to use.
8. The device sensor was easy to wear.
9. This device would be more beneficial with prolonged usage
10. I would use this device in the future.

1	2	3	4	5	6	7	8	9	10