Faculty of Engineering and Technology
Department of Electronics and Communication Engineering
Jain Global Campus, Kanakapura Taluk - 562112
Ramanagara District, Karnataka, India

2018-2022

A Minor Project Report on

"DEMOLITION HAMMER OPERATING TIME RECORDER"

Submitted by

PRAMODH P
18BTREC055

PRANAY KUMAR K
18BTREC056

RAHUL S
18BTREC059

SHATHISH B
18BTREC065

Under the guidance of

Dr. Mohamad Umair Bagali
Assistant Professor
Department of Electronics and Communication Engineering
Faculty of Engineering & Technology
JAIN DEEMED-TO-BE UNIVERSITY

ABSTRACT

Demolition is an essential part of any construction work. Such demolition services are charged on an hourly basis and it is around ₹300 per hour. Usually, these services are provided using an electromechanical tool called “Demolition Hammer”. The problem arises when the demolition service provider, or workers working for such organizations delay the demolition work by not planning their work efficiently. These workers demand charges to be paid for the time in which they are idle. Also due to this reason the work is delayed without any proper reason. This is not an ethical way of doing business. Hence, a timer that is capable of recording the duration of work can induce a significant impact on the work done and making a universal battery operated device will ensure it can be used independent of different brands of demolition hammer.

The project “Demolition Hammer Operating Time Recorder”, aims to solve this problem. The demolition hammer vibrates when it is in operation, this parameter can be taken as a measure to determine whether the work is being done or not. Hence by monitoring the duration the demolition hammer is vibrating, i.e., the work is being done, the services can be charged according to the duration of work. This simple mechanism will force workers to work efficiently and also take breaks only when required. On the other hand, the customer of these service can pay for exact duration of work. Few additional features like ability to store values of last few recordings, user friendly interface, will help the user and the customer.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE USED

LCD	Liquid Crystal Display
DHOTR	Demolition Hammer Operating Time Recorder
LED	Light Emitting Diode
I2C	Inter Integrated Circuit
MEMS	Micro Electrical Mechanical System
EEPROM	Electrically Erasable Programmable Read Only Memory
ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
UART	Universal Asynchronous Receiver Transmitter
SMD	Surface Mount Device
PCB	Printed Circuit Board
IDE	Integrated Development Environment
RISC	Reduced Instruction Set Computer

Chapter 1

1. Introduction

Demolition hammer is an electro-mechanical device that is used to chip and break out concrete, bricks and hard surfaces. These devices are used in construction fields to carry out demolition work. When an individual hires the demolition services, the service is carried out using this hammer and they will be paying the operators of the hammer in hourly basis, the tool which is use at present does not have any device fixed to monitor its activity. Also the cost of current demolition service is around ₹300 per hour. The above case is fine until the users of the hammer, i.e., the works exploit the situation and delay the work. This leads to further delay in following construction work.

Figure 1: Demolition Hammer

Hence we propose our solution for the problem, "Demolition Hammer Time Recorder" it records the total time the machine worked with its vibration. If the demolition hammer is not at work then the timer automatically pauses for that duration and then when the hammer resumes to work then the timer also resumes from that point of time. The proposed device will be able to store few previous time recordings. To have a simple cost effective device, we use a LCD display for user interface. By making use of the proposed device one will be able to measure the exact time of work and can demand payment for the same. We also believe that, incorporating such device in such demolition tasks, we can boost efficiency and ethics of the workers. This will improve the customer satisfaction for those services and accelerate the construction work.

1.1 Literature Survey

In the survey conducted by our project guide Dr. Mohamad Umair Bagali, we have concluded that, many construction workers do not efficiently manage time and delay the work. Since, these services are charged on an hourly basis i.e. ₹300, few workers delay the work unnecessarily and increase the working hours and demand payment for the time they are idle. Hence, some kind of monitoring activity is required to get the worker make the most out of their time. Most of the demolition work is done using this demolition hammer. But instead of a human, a machine can be

precise about the total working time of the workers. If our tool is attached to the demolition hammer it will be able to record time it has been used and workers will be paid for the total time they worked for.

1.2 Limitations of Current Work

The continuous recording time is limited to 18 hours. This limit is due to the size of integer variable of the Arduino C++ compiler. Having the size of integer as 16 bits or 2 bytes can store a maximum value of $2^{16} - 1$ that is 65,535. By dividing 65,535 by 3600 we get 18.20 hours. Since the device is battery operated the contrast and brightness of the LCD display decreases with time, is due to the fact that a potentiometer is used to adjust the contrast and when battery voltage decreases the voltage drop will decrease too. This voltage drop around the potentiometer is used to adjust the contrast, hence the contract decreases along with the voltage drop with time.

1.3 Problem Definition

In this fast progressing world, there is a need for efficiency and speed. But when the construction work's demolition phase is considered, the workers are often found to be inefficient in their work causing unnecessary delays and demanding payment for non-working hours. Also, since construction works depend on demolition services in one or other way it is important to maintain efficiency and ethics in demolition services so that it does not affect the following work. Hence, there arises a need to monitor these workers and thereby completing the work on time. Instead of human supervisor, if an electronic device is used for the same we can achieve accuracy and eliminate any possible errors in supervision.

1.4 Objectives

Design and develop a cost effective add-on time recording device that is capable of recoding the operation time of a demolition hammer. The device should be able to store five previous time recordings and should have an easy to use interface. Also building the device as battery operated, thus making the device universal and able to work independent of manufactures of demolition hammer.

1.5 Methodology

As discussed in the literature review some the hourly based workers do not work efficiently and some are not paid according to their work. So, our DHOTR device can be an aid for this problem. The moto of our project is to get the device to record the operating time of the tool, display the recorded time on the LCD, able to record few previous recordings, could be easily attached to the tool, battery powered and should be easy to interface. The methodology of the device can branched into 4 major segments Timer configuration and Time retrieval, MPU6050 accelerometer configuration and vibration detection, Interfacing of all modules and individual functions required for the overall device and at last Circuit and PCB design. Time Configuration can further be segmented into few steps such as Initializing the 16-bit Timer module of the controller, Make necessary changes for timer's register to make the counter raise an interrupt after each second, For each interrupt increment the "secondRegister" when the vibrations are detected,. For time retrieval Timer and secondRegister is used Consider timer as a free flowing tap, it keeps on counting minutes and stores the value in secondRegister. That time can be obtained by accessing secondRegister at required time. The time is accessed every minute thereby updating the count displayed on the LCD Display, while time is being recorded while the demolition hammer is in operation. Since, the timer is a separate peripheral, its continuous operation does not load the CPU core. The 16 bit register (size of integer) helps in storing value 2^16, which is 65536 minutes and is equivalent to 1092 hours. This is a lot of time for any demolition service. And that is followed be Accelerometer configuration, here some course of actions are underwent which are Configuring MPU6050 with the help of referring to register map provided by the manufacturer, Initialize I2C communication as a master device, Initialize I2C communication as a master device, Configure register(0x6B) and reset the register for configuring the register to normal power setting, Configure register(0x1C) to 0x08 to configure the sensor for +/-4g sensitivity range, End the transmission using Wire endTransmission() method. After initialization of MPU6050 sensor, we can obtain the acceleration, i.e. vibration on each axis, here Vibration come into picture, MPU6050 is an accelerometer along with gyroscope, According to the datasheet, the acceleration data will be stored in a series of six register starting from 0x3B to 0x40.Each axis has 2 8-bit register, a total of 16-bits for each axis's acceleration value. We can read the data stored in these registers through the I2C bus all the registers specified are all present inside in the sensor itself. However these

registers gives us the raw data obtained directly from the MEMS (Micro Electrical Mechanical System) sensor after signal conditioning. We need to process that data in order to obtain the "g" value on the sensor. According to the datasheet for range of +/-4g, the value obtained from sensor is divided by 8192.0 to get the "g" value. Now we have our acceleration or "g" values in our controller, to detect the vibrations we define certain limits in X, Y, Z axis which is experimentally derived. Whenever the acceleration values goes beyond that limits on each axis, a function checkVibration() returns true. This mechanism is used to determine the vibrations and accordingly the microcontroller decides whether to count time or to pause the operation. Interfacing of other Modules excluding the microcontroller unit and the MPU6050 sensor unit are – Status LED (Indicates the status of recording operation. It will be on if the operating time is recorded, and will be off otherwise), Buzzer (Buzzer is on when the device is calibrating, off otherwise), LCD (It is used to display information about all stages of operation), and four push buttons (Used to get user input). The entire circuit is based on ATMEGA328P microcontroller from Atmel (Acquired by Microchip).The reason for choosing this microcontroller is, this microcontroller has a huge community and list of libraries, which make prototyping very fast. This microcontroller has an inbuilt EEPROM of 1KB, so that we need not interface an external EEPROM. This microcontroller has a wide range of operating voltage 2.7 V to 5.5 V @ 16MHz.It is easily available in market as it is a popular microcontroller.

1.6 Hardware and Software Tools used

Table 1: Hardware and Software used

HARDWARE	SOFTWARE
ATMEGA328P Microcontroller	Arduino IDE
MPU6050 Accelerometer cum Gyroscope	Autodesk EAGLE
16*2 LCD Display	
Active Buzzer	
LEDs	
Push Buttons	
Passive Components	

Chapter 2

2. Basic Theory

2.1 Microcontrollers

A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip. A microcontroller is embedded inside of a system to control a singular function in a device. It does this by interpreting data it receives from its I/O peripherals using its central processor. MCUs feature input and output pins to implement peripheral functions. Such functions include analog-to-digital converters, liquid crystal display (LCD) controllers, real-time clock (RTC), universal synchronous/asynchronous receiver transmitter (USART), timers, universal asynchronous receiver transmitter (UART) and universal serial bus (USB) connectivity. Sensors gathering data related to humidity and temperature, among others, are also often attached to microcontrollers.

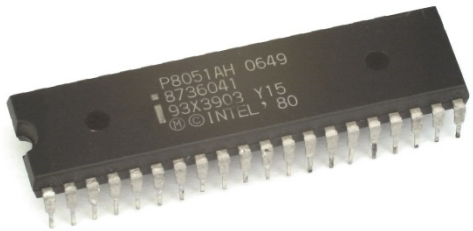

Figure 2: Intel 8051 Microcontroller

The core elements of a microcontroller are CPU, Memory, I/O peripherals and other supporting peripherals include ADC (Analog to Digital Converter), Timers, DAC (Digital to Analog Converter), etc. There are a lot of microcontrollers with diverse architecture to satisfy specific needs of users. For they reduce down to two basic architecture Von Neumann and Harvard. Also considering instruction set as parameter the microcontrollers can be classified into CISC (Complex Instruction Set Computer) and RISC (Reduced Instruction Set Computer) computers. Common MCUs include the Intel MCS-51, often referred to as an 8051 microcontroller, which was first developed in 1985; the AVR microcontroller developed by Atmel in 1996; the programmable interface controller (PIC) from Microchip Technology; and various licensed Advanced RISC Machines (ARM) microcontrollers. These days' microcontrollers have become essential part of our lives. They are hidden in the environment and help up throughout our life.

2.2 Micro Electrical Mechanical System

MEMS is the technology for miniaturized devices which are formed by a combination of electronic as well as mechanical components or elements. Hence, named Micro-electro-mechanical systems. They are obtained from microfabrication techniques. Generally, the size of a MEMS device may vary from one micron to few millimeters. These devices can also vary from no moving structure to complex multiple moving electromechanical structures controlled by integrated microelectronic structures. Although, MEMS are slightly different from nanotechnology but there exist high mutual dependencies between these two technologies.

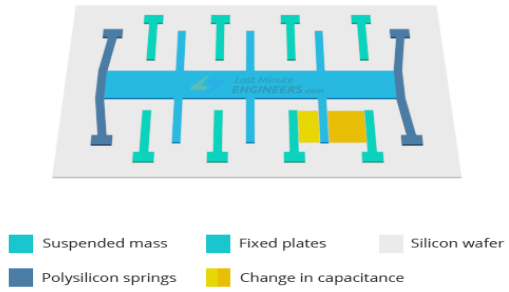

Figure 3: MEMS Accelerometer's internal structure

MEMS accelerometer consists of a micro-machined structure built on top of a silicon wafer. This structure is suspended by polysilicon springs. It allows the structure to deflect at the time when the acceleration is applied on the particular axis. Due to deflection the capacitance between fixed plates and plates attached to the suspended structure is changed. This change in capacitance is proportional to the acceleration on that axis. The sensor processes this change in capacitance and converts it into an analog output voltage. This voltage signal can be processed and acceleration in each axis can be obtained.

2.3 Time Processing

One of the important things is to process the time. Since the project is expected to operate on seconds, it is really important to process it and calculate hours and minutes from the second's data. The reason for using seconds as basic unit of time in our project will be explained in software algorithm. Hence, for conversion we can make use of the following formulae,

For converting seconds to hours,

$$Hours\ from\ Seconds = \frac{Seconds}{3600}$$

For converting seconds to minutes,

$$Minutes\ from\ Seconds = \frac{Seconds\ \%\ 3600}{60}$$

For obtaining remaining seconds,

$$Remaining\ Seconds = Seconds\ \%\ 60$$

2.4 Liquid Crystal Display

A liquid-crystal display (LCD) is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals combined with polarizers. Liquid crystals do not emit light directly, instead using a backlight or reflector to produce images in color or monochrome. LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images with low information content, which can be displayed or hidden. For instance: preset words, digits, and seven-segment displays, as in a digital clock, are all good examples of devices with these displays. They use the same basic technology, except that arbitrary images are made from a matrix of small pixels, while other displays have larger elements. LCDs can either be normally on (positive) or off (negative), depending on the polarizer arrangement. For example, a character positive LCD with a backlight will have black lettering on a background that is the color of the backlight, and a character negative LCD will have a black background with the letters being of the same color as the backlight. Optical filters are added to white on blue LCDs to give them their characteristic appearance.

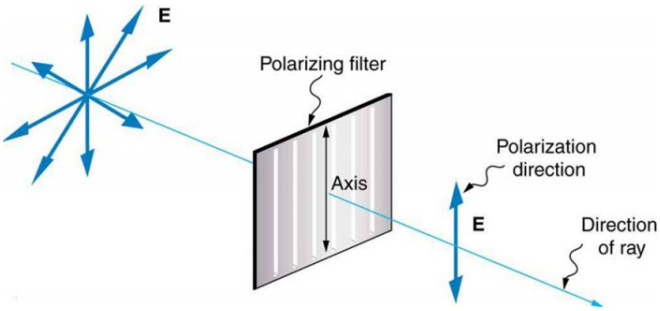

Figure 4: Polarization of light

Chapter 3

3. Tool Description

3.1 ATMEGA328P

The ATmega328 is a single-chip microcontroller created by Atmel in the megaAVR family (later Microchip Technology acquired Atmel in 2016). It has a modified Harvard architecture 8-bit RISC processor core. Atmega328 microcontroller is used in basic Arduino boards i.e. Arduino UNO, Arduino Pro Mini and Arduino Nano. This Atmel 8-bit AVR RISC-based microcontroller combines 32 KB ISP flash memory with read-while-write capabilities, 1 KB EEPROM, 2 KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts. The device achieves throughput approaching 1 MIPS per MHz

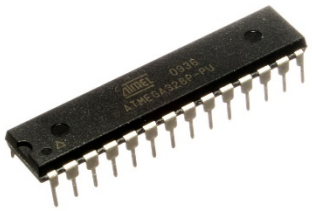

Figure 5: ATMEGA328P Microcontroller

3.2 MPU6050

The MPU-6050 is the world's first Motion Tracking devices designed for the low power, low cost, and high-performance requirements of smartphones, tablets and wearable sensors. The MPU-6050 incorporates InvenSense's MotionFusion™ and run-time calibration firmware that enables manufacturers to eliminate the costly and complex selection, qualification, and system level integration of discrete devices in motion-enabled products, guaranteeing that sensor fusion algorithms and calibration procedures deliver optimal performance for consumers. The MPU-

Figure 6: MPU6050 Accelerometer cum Gyroscope

6050 devices combine a 3-axis gyroscope and a 3-axis accelerometer on the same silicon die, together with an onboard Digital Motion Processor (DMP), which processes complex 6-axis MotionFusion algorithms. The device can access external magnetometers or other sensors through an auxiliary master I²C bus, allowing the devices to gather a full set of sensor data without intervention from the system processor. The devices are offered in a 4 mm x 4 mm x 0.9 mm QFN package. This device is based on MEMS (Micro Electrical Mechanical System) for its operation. Let's look at MEMS in the following sub division.

3.3 LCD Display

Liquid crystal display (LCD), electronic display device that operates by applying a varying electric voltage to a layer of liquid crystal, thereby inducing changes in its optical properties. LCDs are commonly used for portable electronic games, as viewfinders for digital cameras and camcorders, in video projection systems, for electronic billboards, as monitors for computers, and in flat-panel televisions. Liquid crystals are materials with a structure that is intermediate between that of liquids and crystalline solids. As in liquids, the molecules of a liquid crystal can flow past one another. As in solid crystals, however, they arrange themselves in recognizably ordered patterns. In common with solid crystals, liquid crystals can exhibit polymorphism; i.e., they can take on different structural patterns, each with unique properties. LCDs utilize either nematic or smectic liquid crystals.

Figure 7: 16X2 LCD Display

3.4 Interfacing and Passive Components

3.4.1 Active Buzzer

An active buzzer will generate a tone using an internal oscillator, so all that is needed is a DC voltage. A passive buzzer requires an AC signal to make a sound. It is like an electromagnetic speaker, where a changing input signal produces the sound, rather than producing a tone automatically.

Figure 8: Active Buzzer

3.4.2 LED

A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons. The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the band gap of the semiconductor. White light is obtained by using multiple semiconductors or a layer of light-emitting phosphor on the semiconductor device.

Figure 9: LEDs

3.4.3 Passive Components

Passive electronic components are those that don't have the ability to control electric current by means of another electrical signal. Examples of passive electronic components are capacitors, resistors, inductors, transformers, and some diodes.

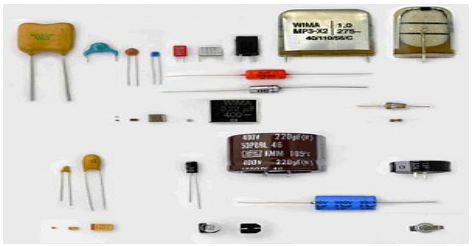

Figure 10: Passive Components

3.5 Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them. Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension ".ino". The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving

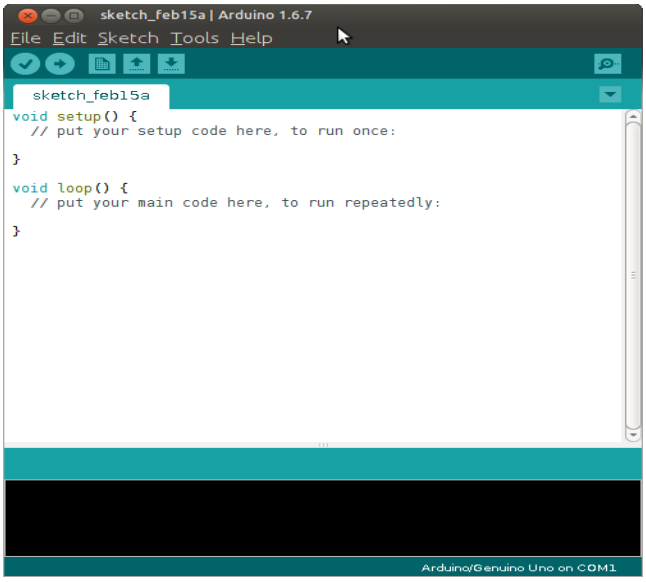

Figure 11: Arduino IDE

and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

3.6 Autodesk EAGLE

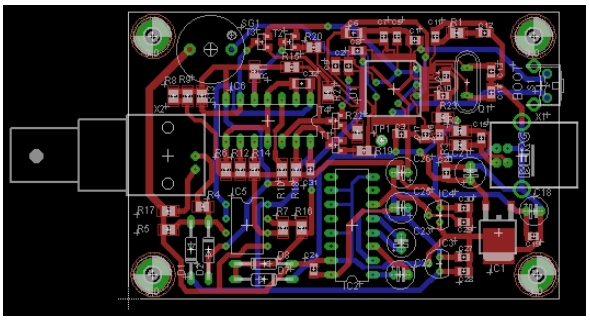

Figure 12: Screenshot of Autodesk EAGLE

Autodesk EAGLE is a scriptable electronic design automation (EDA) application with schematic capture, printed circuit board (PCB) layout, auto-router and computer-aided manufacturing (CAM) features. EAGLE stands for Easily Applicable Graphical Layout Editor (German: Einfach Anzuwendender Grafischer Layout-Editor) and is developed by CadSoft Computer GmbH. The company was acquired by Autodesk Inc. in 2016.

Chapter 4

4. Implementation

4.1 Hardware Design and Implementation

4.1.1 Schematic

The schematic of the device is as follows,

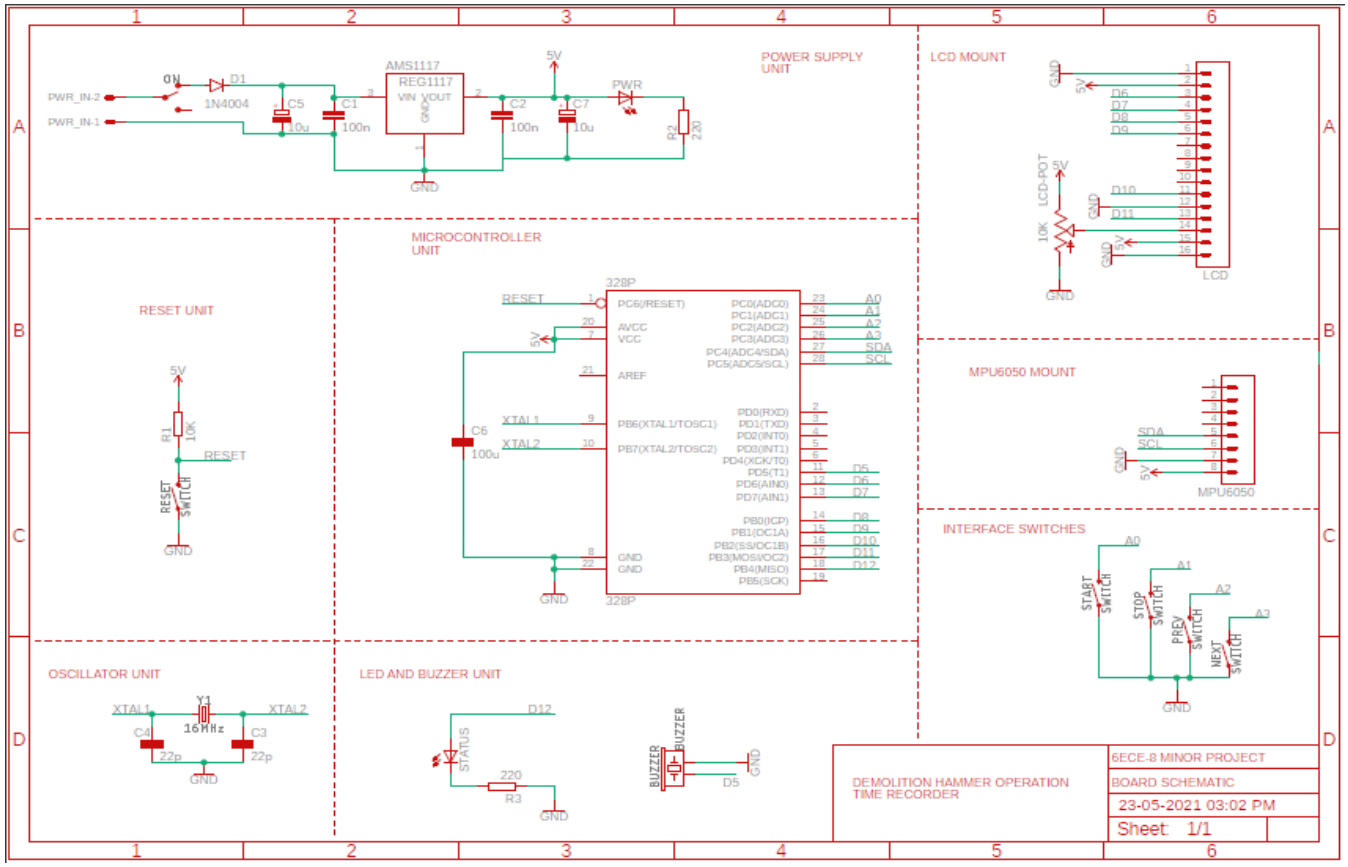

Figure 13: Schematic

4.1.2 PCB Design

The PCB (Printed Circuit Board) for the device is a 2 layer board of 63 mm * 51 mm in dimension. The layout for the PCB has been created with Autodesk EAGLE. The PCB component placement

process involved multiple iterations. Using trial and error method the best possible PCB layout and component placement was obtained. It is mentioned as follows,

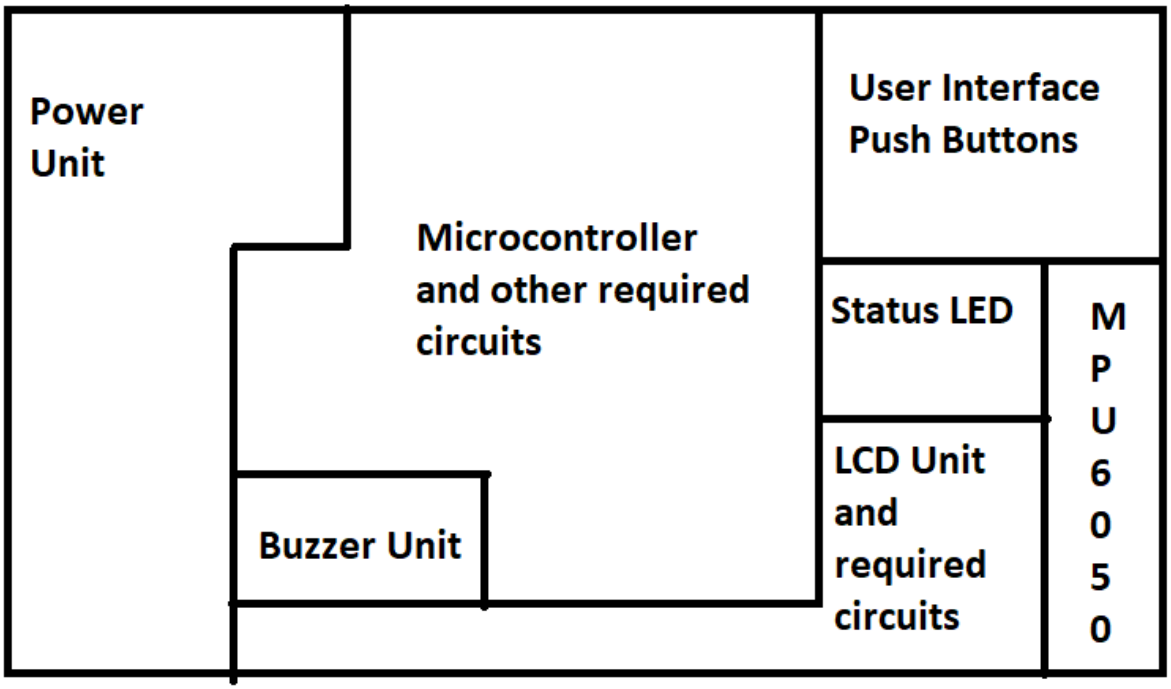

Figure 14: PCB Component Placement Layout

And the PCB is as follows,

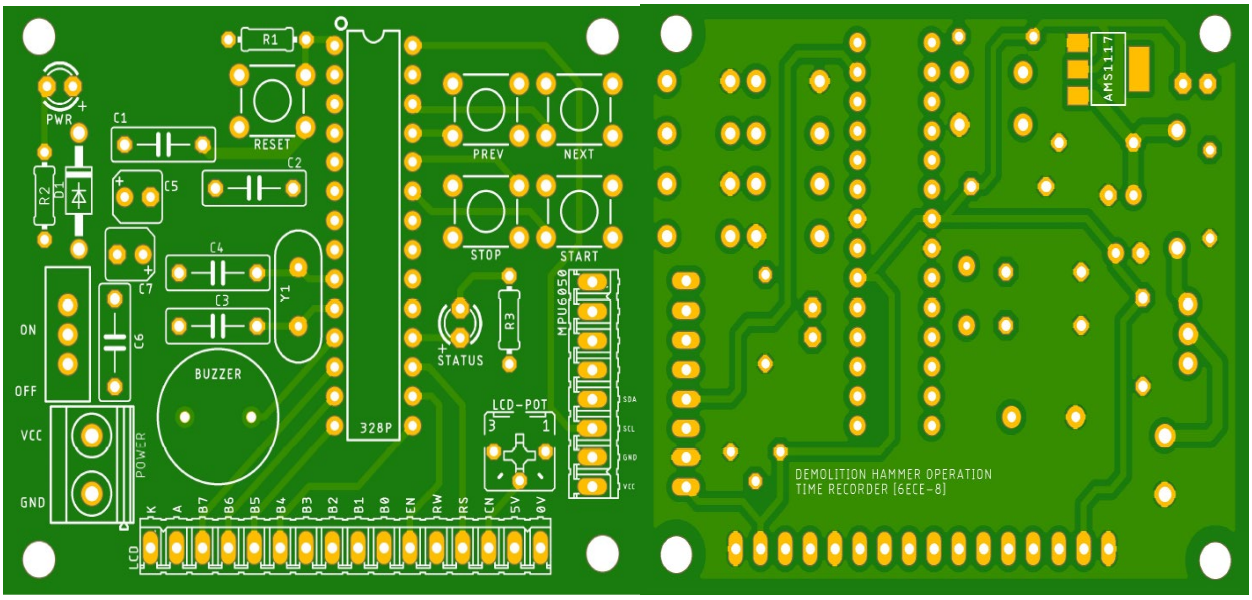

Figure 15: PCB Top (Left) and Bottom (Right)

4.2 Software algorithm

4.2.1 Timer configuration and operation

To initialize the 16-bit Timer module of the controller, we need to make necessary changes timer's register to make the counter raise an interrupt after each second. For each interrupt increment the "secondRegister" when the vibrations are detected. Hence we have an interrupt every second and only when vibration is detected, the secondRegister is incremented and the overall duration of the timer increases. Consider timer as a free flowing tap, it keeps on counting seconds and stores the value in secondsRegister. That time can be obtained by accessing secondRegister at required time. The time is accessed every second thereby updating the count displayed on the LCD Display, while time is being recorded while the demolition hammer is in operation. Since, the timer is a separate peripheral, its continuous operation does not load the CPU core. The 16 bit register (size of integer) helps in storing value up to 2^16-1, which is 65536 seconds and is equivalent to 18 hours. This is a lot of time for any demolition service.

4.2.2 MPU6050 Configuration and Vibration Detection

Configuring MPU6050 is done by referring to register map provided by the manufacturer. First, initialize I2C communication as a master device. Initially the MPU6050's slave I2C address is passed as an argument to Wire.beginTransmission() method to initialize I2C communication. Then, configure register (0x6B) and reset the register for configuring the register to normal power setting and configure register (0x1C) to 0x08 to configure the sensor for +/-4g sensitivity range. After the configuration is over end the transmission using Wire.endTransmission() method.

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B|	59	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT[7:0]							

Table 2: Register values from MPU6050 Register Map

After initialization of MPU6050 sensor, we can obtain the acceleration, i.e. vibration on each axis. Here MPU6050 is an accelerometer along with gyroscope but we don't make use of the gyroscope.

According to the datasheet, the acceleration data will be stored in a series of six register starting from 0x3B to 0x40. Each axis has 2 8-bit register, a total of 16-bits for each axis's acceleration value. We can read the data stored in these registers through the I2C bus. Note: These registers mentioned above are inside the sensor itself.

However these registers gives us the raw data obtained directly from the MEMS (Micro Electrical Mechanical System) sensor after signal conditioning. We need to process that data in order to obtain the "g" value on the sensor. According to the datasheet for range of +/-4g, the value obtained from sensor is divided by 8192.0 to get the "g" value. Now we have our acceleration or "g" values in our controller, to detect the vibrations we define certain limits in X, Y, Z axis which is experimentally derived. Whenever the acceleration values goes beyond that limits on each axis, a function checkVibration() returns true. This mechanism is used to determine the vibrations and accordingly the microcontroller decides whether to count time or to pause the operation.

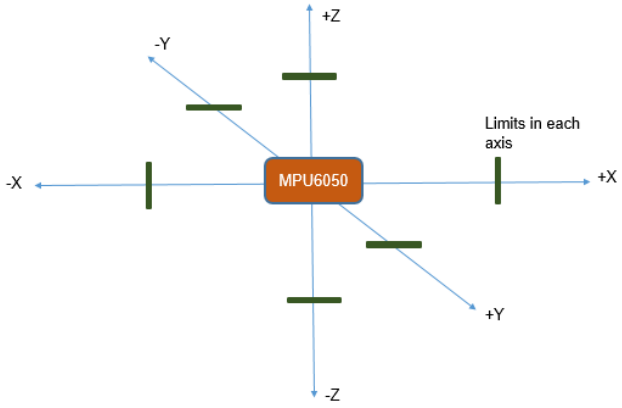

Figure 16: Value limits in each axis to detect vibration

4.2.3 EEPROM Operations

Also there is EEPROM (Electrically Erasable Programmable Read Only Memory) inside the microcontroller itself. This Non-Volatile memory is useful in storing the values of last recordings as mentioned in features list. These values can be accessed by LCD display of the device. The EEPROM is 1KB wide and stores data as 8-bit chucks. But the size of integer we use is 16-bits hence, the data is broken into two pieces of 1 byte (8-bits) each. The first half is stored in even addresses of EEPROM and second half is stored in odd addresses of EEPROM alternatively. That stored data is retrieved in the same way. For splitting the data into higher 8 bits and lower 8 bits, it is multiplied with 0xFF00 and 0x00FF respectively. Then for retrieving the data both multiplied with 0xFFFF and ORed with each other.

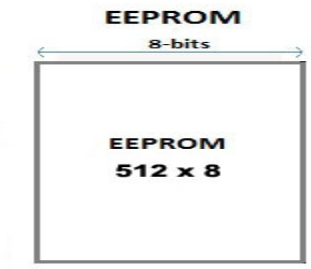

Figure 17: Memory Map of EEPROM in ATMEGA328P

4.2.4 Interfacing Other Modules

The interfacing of other modules include interfacing of buzzer, status LED, LCD and four push buttons namely “START”, “STOP”, “NEXT”, and “PREV”. The algorithm involved is summarized in the following flow chart,

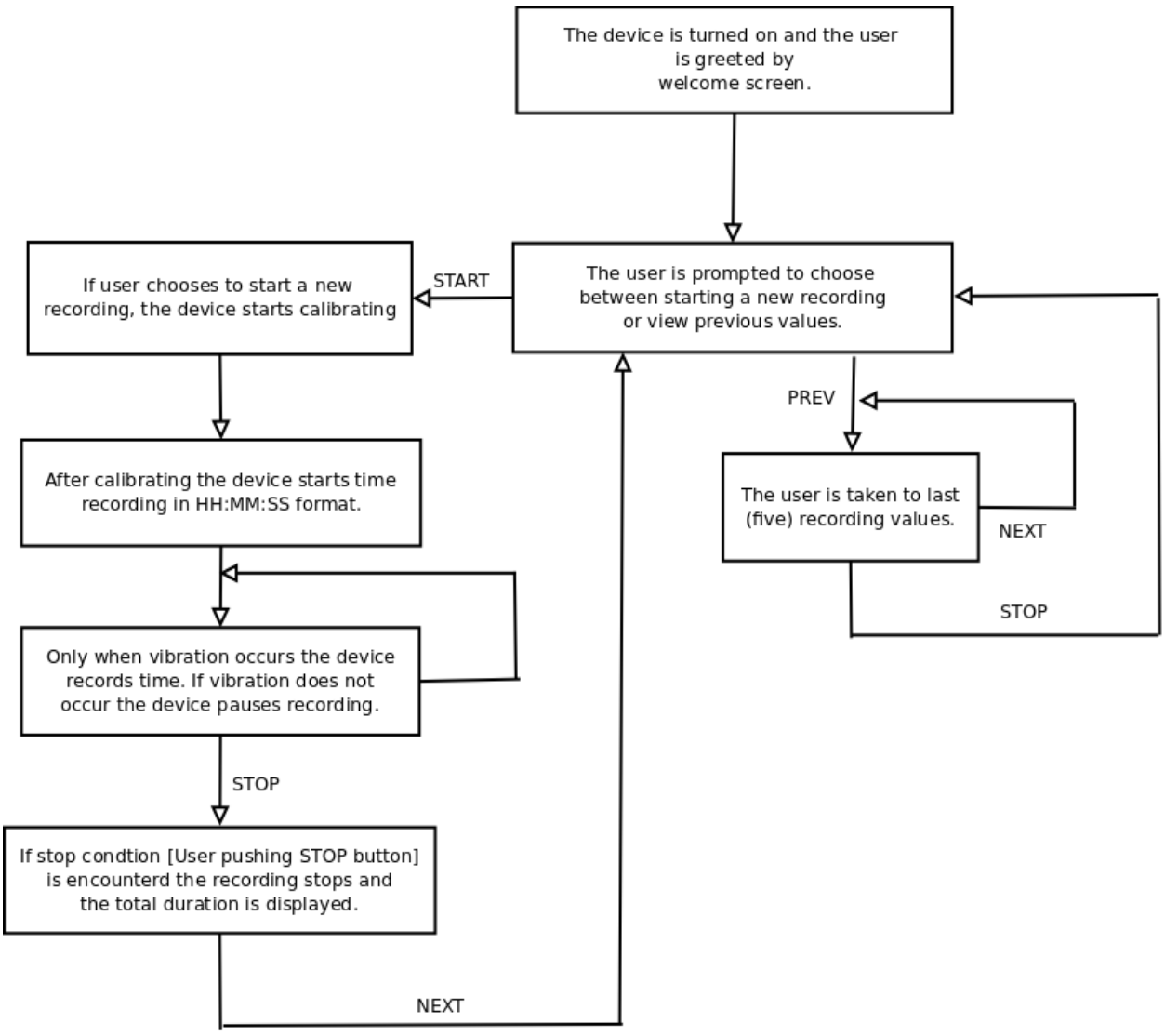

Figure 18: Basic Operation Flowchart

Chapter 5

5. Results and Discussion

As mentioned earlier DHOTR (Demolition Hammer Operating Time Recorder) device is an add-on device to a Demolition hammer which records the operation time with the machines vibration. The device operation is explained below. The final device is,

Figure 19: Demolition Hammer Operating Time Recorder

5.1 Boot Up

The device is turned on using a slide switch. After the device turns on the device displays “DHOTR Minor Project 6ECE-8” on the LCD screen. Then the user is redirected to the selection screen.

Figure 20: Title Screen

5.2 Selection

In the selection screen the user can choose between starting a new recording and viewing previous values. If user chooses to view values of previous time recordings, then user will be redirected to previous value screen. If the user chooses to start a new recording, then user is redirected to time recording procedure.

Figure 21: Selection Screen

5.3 Time Recording

When the user chooses to start a new recording, the calibration process starts. In the calibration process, the timer module is configures, MPU6050 is configured and calibrated, global interrupts are enabled and the EEPROM shift operation is completed to accommodate the new recording value into the EEPROM. After the calibration, the device starts recording the duration of operation. The timer paused accordingly when no vibration occurs i.e., when user is not using the demolition hammer.

Figure 22: Device Calibration

Figure 23: Device Calibration completed

Figure 24: Time Recording

Figure 25: Time Recording Completed

After the work is completed the user can press “STOP” push button and review the total duration of work. Then when user presses “NEXT”, is redirected to the selection screen.

5.4 Previous Value

If the user chooses to view the values of previous recordings, user is redirected to Previous values screen. Here the first value labeled “P5” is the value of latest recording. If user presses “NEXT”, is redirected to “P4” which is second latest value. This continues till the user reaches “P1” which is fifth latest recording value. If the user wishes to exit to selection screen after viewing a value, then user can press “STOP”.

Figure 26: Previous Value Screen

Conclusions and Future Scope

To conclude sometimes demolition hammer users are paid more than they are required or the workers are not paid for their working hour, to satisfy the payer or payee we can use our device DHOTR which can be connected to the demolition hammer and record the time for how many hours or minutes can be used. Our device has a limit of 18 hour recording time with few of upgrades this can improved, for example with few changes in the program we can increase the time duration, and with the help of SMD components we can decrease the dimension of the device.

Also this project can be presented to industries which manufacture demolition hammers. We can apply as a consultant and propose this project to be part of their product itself instead of being an add-on device.

REFERENCES

[1] (2021) “Arduino Reference”. [Online]. Available: https://www.arduino.cc/reference/en/

[2] “MPU-6000 and MPU-6050 Product Specification”, Revision 3.4, InvenSense Inc., USA.

[3] “MPU-6000 and MPU-6050 Register Map and Descriptions”, Revision 4.2, InvenSense Inc., USA.

[4] “AMS1117 Datasheet”, KEXIN.

[5] “Atmega.48A/PA/88A/PA/168A/PA/328/P megaAVR® Data Sheet”, Microchip.

APPENDIX - I

SOURCE CODE

```
/*****************************************************************************
  Arduino Code for "DEMOLITION HAMMER OPERATING TIME RECORDER"
  Minor Project - 18EC66
  Team - 6ECE-8
  Project Guide - Dr.Mohmad Umair Bagali
  Team Members - Shatish B [18BTREC065]
                 Pramodth P [18BTREC055]
                 Rahul S [18BTREC059]
                 Pranay Kumar K [18BTREC056]

  Current compiled binary size: 10576 Bytes
*****************************************************************************/
// Includes
#include<Wire.h>
#include<EEPROM.h>
#include<LiquidCrystal.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#define MPU 0x68
#define MAX_X 0.1
#define MAX_Y 0.1
#define MAX_Z 0.1

// Object initilization
LiquidCrystal lcd(11, 10, 9, 8, 7, 6);

// Variable definitions
int secondRegister = 0;
float AccX, AccY, AccZ;
float x, y, z;
// Switches
int START = A2;
int STOP = A1;
int PREV = A0;
int NEXT = A3;
// Buzzer
int Buzzer = 5;
// Status LED
int statusLED = 12;
```

```
int doesVibrate;
int *doesVibratePtr = &doesVibrate;

// Fuction Declarations
void writeToEEPROM(int addr, int value);
int readFromEEPROM(int addr);
void initMPU6050();
void checkVibration();
void initTimer();
void titleScreen();
void mainMenu();
void calibrating();
void previousValue(int addr);
void calibratingDone();
void displayCurrentTime();
void finalTime();
void shiftEEPROM();
String returnHRTime(int secondRegister);

void setup() {
  Serial.begin(115200);
  lcd.begin(16, 2);
  pinMode(START, INPUT_PULLUP);
  pinMode(STOP, INPUT_PULLUP);
  pinMode(PREV, INPUT_PULLUP);
  pinMode(NEXT, INPUT_PULLUP);
  pinMode(Buzzer, OUTPUT);
  pinMode(statusLED , OUTPUT);
  titleScreen();
  delay(3000);
}

void loop() {
  mainMenu();
  while (1) {
    if (digitalRead(START) ^ digitalRead(PREV)) {
      if (digitalRead(START)) {
        calibrating();
        digitalWrite(Buzzer, HIGH);
        initTimer(); // Initilize Timer
        initMPU6050(); // Initilize MPU6050
        shiftEEPROM();
        delay(2000);
        digitalWrite(Buzzer, LOW);
```

```
        digitalWrite(statusLED, HIGH);
        calibratingDone();
        delay(1000);
        while (1) {
          checkVibration();
          displayCurrentTime();
          delay(500);
          if (!digitalRead(STOP)) {
            digitalWrite(statusLED, LOW );
            cli();
            finalTime();
            while (digitalRead(NEXT));
            secondRegister = 0;
            break;
          }
        }
        break;
      }
      if (digitalRead(PREV)) {
        int count = 4;
        while (1) {
          if ((count > 4) | (count < 0)) {
            break;
          }
          else {
            previousValue(count);
            while (!(digitalRead(STOP) ^ digitalRead(NEXT)));
            if (!digitalRead(NEXT)) {
              count--;
              delay(2000);
            }
            else if (!digitalRead(STOP)) {
              break;
            }
          }
        }
        break;
      }
    }
  }
}

// Function Definitions
void writeToEEPROM(int addr, int value) {
```

```
  byte first = (0XFF00 & value) >> 8;
  EEPROM.update(addr * 2, first);
  byte sec = 0X00FF & value;
  EEPROM.update((addr * 2) + 1, sec);
}

int readFromEEPROM(int addr) {
  return (0XFFFF & (EEPROM.read(addr * 2) << 8)) | (EEPROM.read((addr * 2) + 1));
}

void initTimer() {
  cli(); // Disable global interrupt
  // Configuring Timer
  // Clearing Bits to clear Garbage values in registers
  TCCR1B = 0x00;
  TCCR1A = 0x00;
  // Starting Timer with Prescalar as 1024 (CSxx - for selecting prescalar)
  // WGM12 - used in mode 4 for using CTC mode (Clear Timer Capture mode)
  TCCR1B = (1 << CS12) | (0 << CS11) | (1 << CS10) | (1 << WGM12);
  // Initialize counter
  TCNT1 = 0;
  // Setting up TCNT1 to compare with 62500 on OCR1A value determined using formula
  OCR1A = 15625;
  // Enable interupt for compare
  TIMSK1 = (1 << OCIE1A);
  sei(); // Enable global interupt
}

void initMPU6050() {
  Wire.begin();                 // Initialize comunication
  Wire.beginTransmission(MPU);      // Start communication with MPU6050 // MPU=0x68
  Wire.write(0x6B);               // Talk to the register 6B
  Wire.write(0x00);               // Make reset - place a 0 into the 6B register
  Wire.endTransmission(false);
  Wire.write(0x1C);               // Talk to the register 1C
  Wire.write(0x08);               // Write 0X08 to select +-4g for range
  Wire.endTransmission(true);       //end the transmission
  delay(20);
  Wire.beginTransmission(MPU);
  Wire.write(0x3B); // Start with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true); // Read 6 registers total, each axis value is stored in 2 register
s
  //For a range of +-4g, we need to divide the raw values by 8192, according to the datasheet
```

```
  x = abs((Wire.read() << 8 | Wire.read()) / 8192.0); // X-axis value
  y = abs((Wire.read() << 8 | Wire.read()) / 8192.0); // Y-axis value
  z = abs((Wire.read() << 8 | Wire.read()) / 8192.0); // Z-axis value
}

void checkVibration() {
  Wire.beginTransmission(MPU);
  Wire.write(0x3B); // Start with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true); // Read 6 registers total, each axis value is stored in 2 register
s
  //For a range of +-4g, we need to divide the raw values by 8192, according to the datasheet
  AccX = (Wire.read() << 8 | Wire.read()) / 8192.0; // X-axis value
  AccY = (Wire.read() << 8 | Wire.read()) / 8192.0; // Y-axis value
  AccZ = (Wire.read() << 8 | Wire.read()) / 8192.0; // Z-axis value
  if (AccX < 0) {
    AccX = abs(AccX);
  }
  if (AccY < 0) {
    AccY = abs(AccY);
  }
  if (AccZ < 0) {
    AccZ = abs(AccZ);
  }
  if (((AccX - x) > MAX_X) | ((AccY - y) > MAX_Y) | ((AccZ - z) > MAX_Z)) {
    x = AccX;
    y = AccY;
    z = AccZ;
    *doesVibratePtr = true;
  }
  else {
    x = AccX;
    y = AccY;
    z = AccZ;
    *doesVibratePtr = false;
  }
}

void titleScreen() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("DHOTR - 6ECE-8");
  lcd.setCursor(0, 1);
  lcd.print("Minor Project");
```

```
}

void mainMenu() {
  lcd.clear();
  lcd.setCursor(3, 0);
  lcd.print("Choose One");
  lcd.setCursor(0, 1);
  lcd.print("START      PREV");
}

void calibrating() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Calibrating...");
  lcd.setCursor(0, 1);
  lcd.print("Do not move");
}

void previousValue(int addr) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("P");
  lcd.setCursor(1, 0);
  lcd.print(addr + 1);
  lcd.setCursor(2, 0);
  lcd.print(" - ");
  lcd.setCursor(5, 0);
  lcd.print(returnHRTime(readFromEEPROM(addr)));
  lcd.setCursor(0, 1);
  lcd.print("STOP       NEXT");
}

void calibratingDone() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Done. Time");
  lcd.setCursor(0, 1);
  lcd.print("Rec. Started");
}

void displayCurrentTime() {
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("Time Elapsed");
```

```
  lcd.setCursor(4, 1);
  lcd.print(returnHRTime(secondRegister));
}

void finalTime() {
  lcd.clear();
  lcd.setCursor(1, 0);
  lcd.print("Total Duration");
  lcd.setCursor(0, 1);
  lcd.print(returnHRTime(secondRegister));
  lcd.setCursor(12, 1);
  lcd.print("NEXT");
}

void shiftEEPROM() {
  int i;
  for (i = 0; i < 4; i++) {
    writeToEEPROM(i, readFromEEPROM(i + 1));
  }
}

String returnHRTime(int secondRegister) {
  int hh = secondRegister / 3600;
  int mm = (secondRegister % 3600) / 60;
  int ss = secondRegister % 60;
  String temp = "";
  if (hh <= 9) {
    temp.concat("0");
  }
  temp.concat(hh);
  temp.concat(":");
  if (mm <= 9) {
    temp.concat("0");
  }
  temp.concat(mm);
  temp.concat(":");
  if (ss <= 9) {
    temp.concat("0");
  }
  temp.concat(ss);
  return temp;
}

//Declare and Define ISR for CTC Interrupt
```

```
ISR(TIMER1_COMPA_vect) {
  if (doesVibrate) {
    secondRegister++;
    writeToEEPROM(4, secondRegister);
  }
  TCNT1 = 0x00;
}
```