REPORT

Abstract

Demolition hammer is a electro-mechanical device that is used to chip and break out concrete, bricks and hard surfaces. Which are used in construction fields mostly. When a individual hire the demolition services they use this hammer and they are paying the operates in hourly basis, the tool which is used at present does not have a timer fixed to it and the cost of current demolition service per hour is approximately INR 300.

My project “Demolition Hammer Time Recorder” is useful for concrete demolition and it records the total time the machine worked with its vibration. If the tool goes to rest mode then the timer automatically pauses for some time and then when the machine resumes to work then the timer also resumes from that point of time. This is able to store few previous recordings. Displays the recorded timings on LCD display. And our project’s main goal is to create “easy user interface”.

Requirements

Hardware

- ATMEGA328P Microcontroller
- MPU6050 Accelerometer
- 16*2 LCD Display
- And other interfacing components

Software

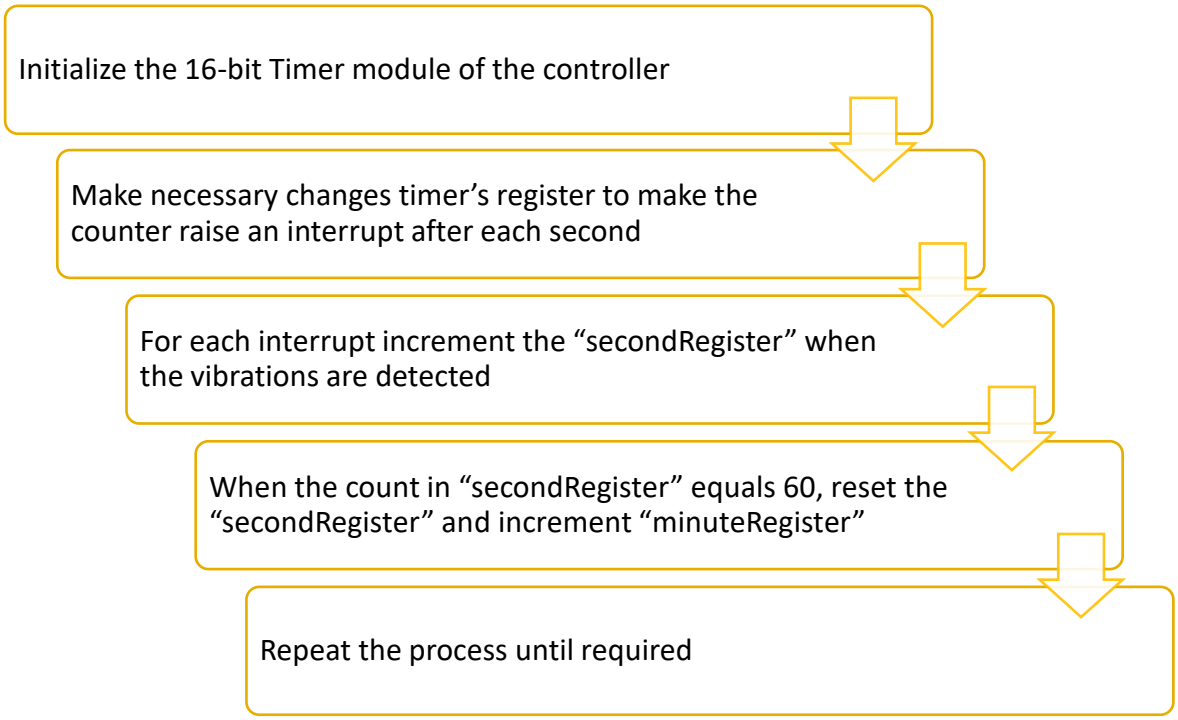
- Arduino IDE (Programming and Burning)
- Autodesk EAGLE (PCB Design)
- SimulIDE (Simulation)

Testing Technologies

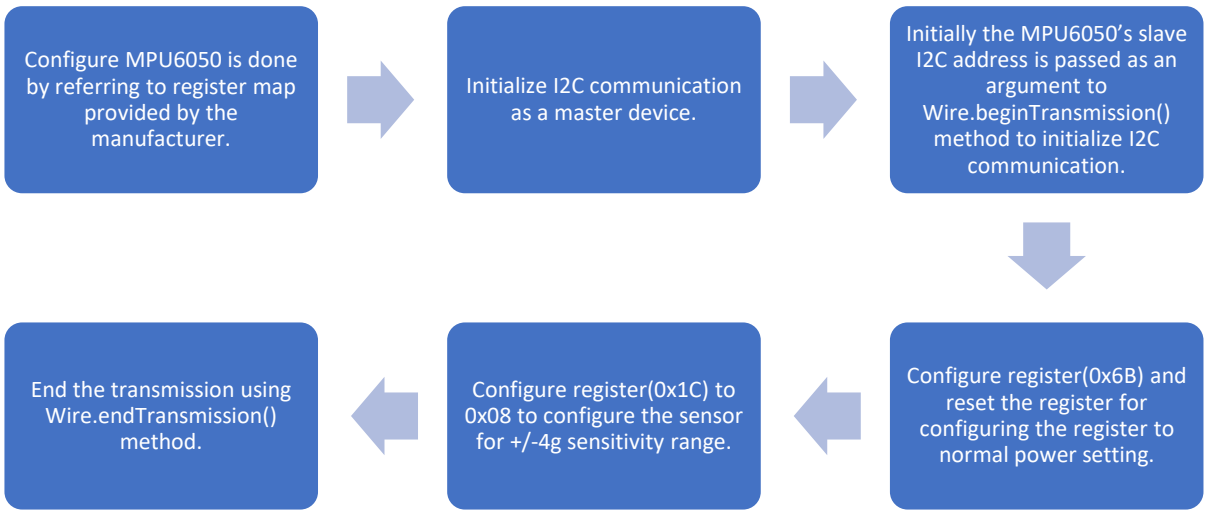
- Multimeter
- GrebV (Gerber file viewer)
- Arduino UNO Development board
- Breadboard and other passive components

Design

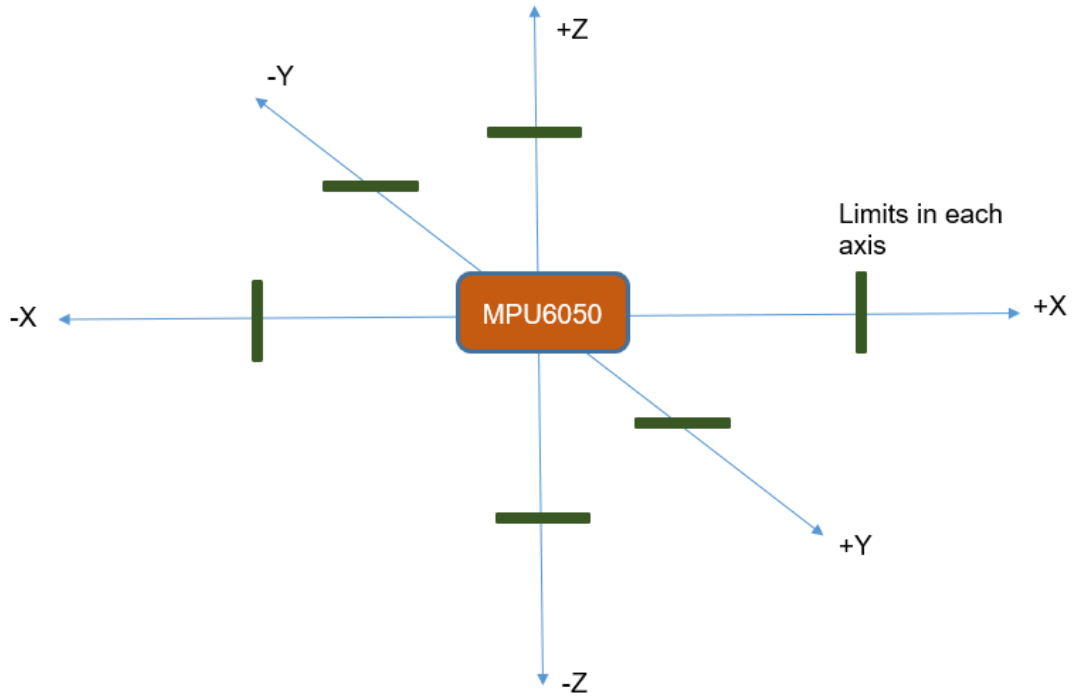
Timer Configuration



Accelerometer Configuration



Detection of Vibrations



Basic Operation Flow

The user turns the device ON using the slide switch.

The user is welcomed by starting screen on LCD, “DHOTR Minor Project”.

In the following screen the user can choose from viewing previous recordings or can start a new recoding.

If user chooses to view previous recordings, the user will be shown last five recoded times.

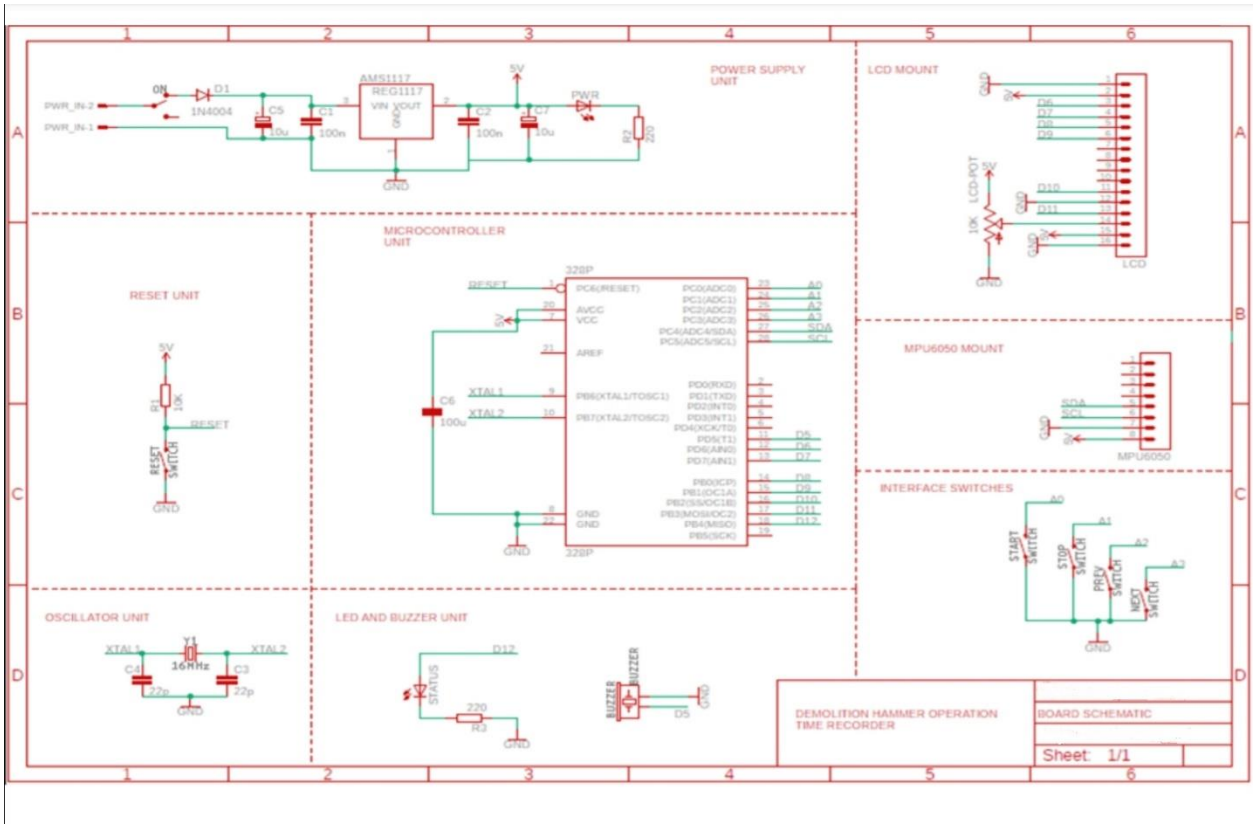
If user chooses to start a new recording, the device starts calibrating and the buzzer plays a tone.

After calibrating, the recoding is started. The green status LED is on and buzzer turns off. The time which is being reordered is shown on LCD screen.

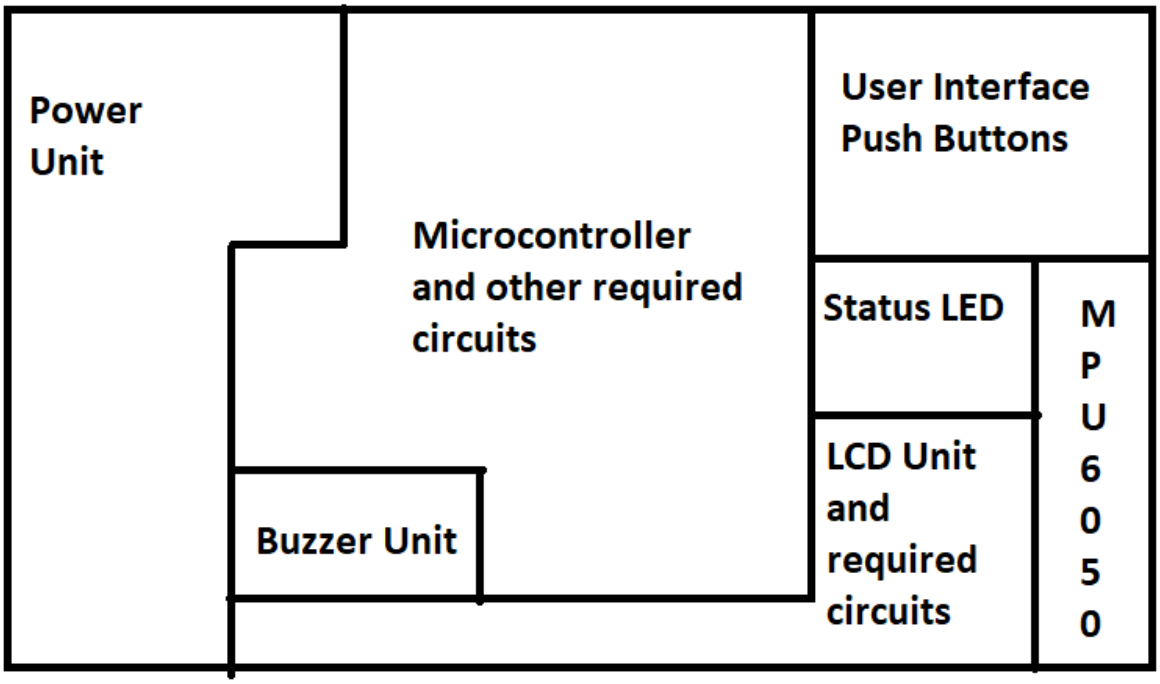
After completion the work, user can stop the recording, this value is displayed and also stored in EEPROM.

Then, after viewing the value and pressing prompted button, user will be redirected to the main screen.

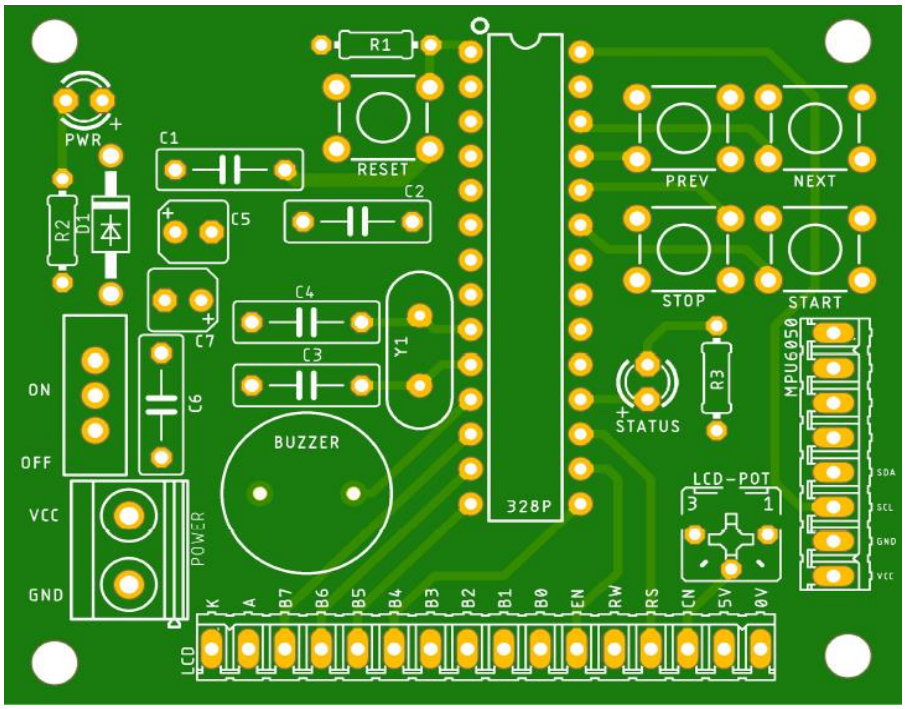
Schematic



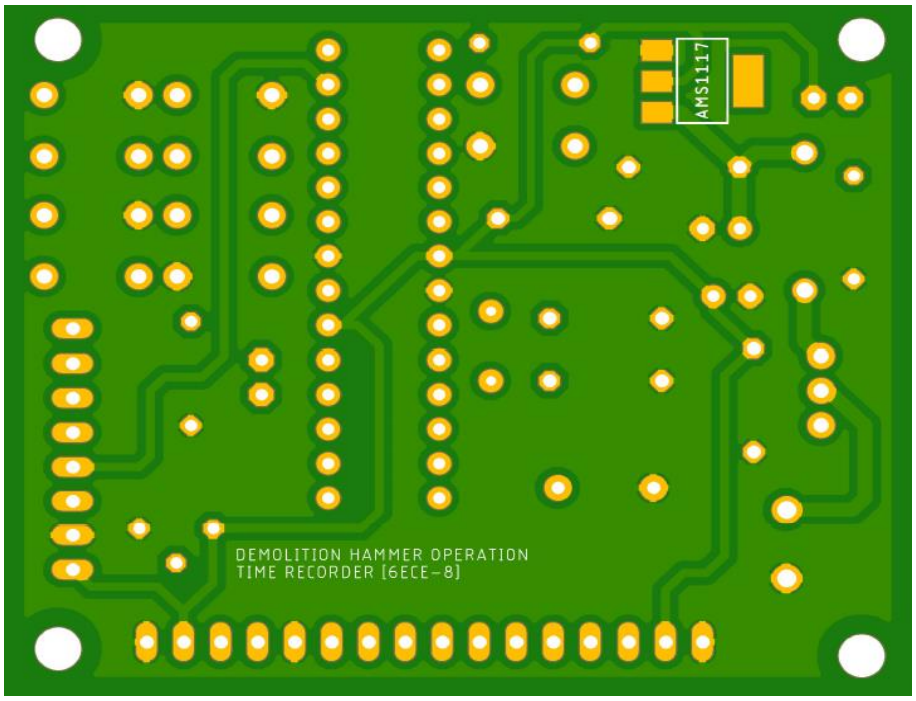
PCB Component Placement



PCB [Top]



PCB [Bottom]



Implementation

```
#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 fo
rmula
  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 // MP
U=0x68
  Wire.write(0x6B);                  // Talk to the register 6B
  Wire.write(0x00);                  // Make reset - place a 0 into the 6B reg
ister
  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 i
s stored in 2 registers
  //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 i
s stored in 2 registers
  //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;
}
```

Test Cases

The final device will have the following features,

Record the operating time of demolition hammer.

Display the recorded time on LCD display.

Battery operated.

Able to attach itself to the demolition hammer (Can make the device universal).

Easy user interface.

Ability to store recording for few previous recordings.

The entire project is divided into four major sections,

Timer configuration and time retrieval.

MPU6050 Accelerometer configuration and vibration detection.

Interfacing of all modules and individual functions required for the overall device.

Circuit and PCB design.