**FIRST QUARTER DESIGN PROJECT - FINAL REPORT**

***Fall-Meter***

**EE 111/151 - Fall 2020**

*A close up of a device

Description automatically generated*

*Design Engineer:* ***Jacqueline Radding***

**EE 111/151 – First Quarter Design Project**

Project Design - Final Specification

**Project Name: FallMeter**

**Project Goal:**

|  |
| --- |
| The FallMeter alerts Senior center workers when a patient falls. Sometimes, elderly patients fall and cannot press a fall button if they have a medical emergency. The FallMeter is a wireless Bluetooth solution that protects the elderly and alerts staff if medical attention is needed to a main computer or tablet. Senior centers are understaffed due to COVID-19 and need a way too check-up on their patients. The FallMeter alerts staff on the main tablet when they are online. If a patient falls, an alert will be sent to the staff and a sound will be a played on a buzzer (which can be changed by a potentiometer), and bright LED will turn on to alert those around the patient. A patient can also click an emergency button on the FallMeter also. Since many fallen senior patients cannot click a fall button necklace if they have a medical emergency or a broken arm. |

**Intended Customer / User:** Who is most likely to want to use your project? Any specific age groups, or geographic locations, or particular occupations who might be more likely to use it?

|  |
| --- |
| Senior centers will provide their patients with these devices. It will be used by individuals aged sixty and up. |

**Product Use Environment:** Will your project be used indoors or outdoors; in a home, office, factory or other setting?

|  |
| --- |
| Indoor use in a senior center or hospital.  Temperature Range: 50 oF – 85 oF |

**Market Research:**

|  |
| --- |
| There are push fall buttons that are $20 and smart watches such as the Apple Watch that are $300-$400 that have fall detection. |

**Customer Needs: Features and Functions**

|  |
| --- |
| * **Bluetooth/wireless fall detection** * **Bluetooth alert interface on a tablet** * **Buzzer alert with a pitch that can be altered by a potentiometer** * **An emergency button that sends an alert** * **Acceleration calibration** * **LED fall indicator** * **Vigorous acceleration/fall alert and lighter fall counter** |

**Performance Requirements?**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Specification** | **Requirement** | **Units** | | Fall Detection Response Time | Under 3 seconds to send alert | seconds | |  |  |  | |  |  |  | |

**Project Learning Requirements Checklist:**

**Rating: 1 –Definitely Does Not 2 –Not Sure If It Meet or Not 3 –Meets the Requirement 5 –Goes Beyond the Requirement**

|  |  |
| --- | --- |
| **Project Learning Requirements** | **Rating** |
| 1. Use a library (need functions someone else already wrote for you) | **5** |
| 1. Use lots of different programming techniques | **5** |
| 1. Use a hardware interrupt feature | ***3*** |
| 4a. Use at least one sensor as an input | **5** |
| 4b. Control something that is variable (like motor speed, or brightness) | **5** |
| 4c. Control something that is ON/OFF | **5** |
| 1. Includes some wireless or remote-control feature | **3** |
| 1. Incorporates a display (like the LCD text display) or sends info back to the computer using the serial connection (USB cable) | **5** |
| 1. Integrates at least one device, sensor, or command that was not included in EE 151 projects | **5** |
| 7a. What is it?: | **HC-05 Bluetooth module and**  **GY-521 Accelerometer** |
| 1. I wrote well-documented, easy-to-read Arduino code. | **5** |

**User Interface:**

**Inputs:**

|  |
| --- |
| * **X,Y, and Z Acceleration/ Acceleration Magnitude, Potentiometer Resistance, Emergency button** |

**Outputs:**

|  |
| --- |
| * **LED On, Fall Counter, Buzzer frequency, calibrated acceleration, Bluetooth Fall Alert** |

**Hardware Design**

**Diagram, schematic

Description automatically generated**

**Diagram, schematic

Description automatically generated**

**Software Design**

**//FALLMETER**

**//BY JACQUELINE RADDING**

**/\* FALLMETER is a wireless device that detects a fall or pressed emergency button and sends the recent fall alert and data to a bluetooth connected device. \*/**

**#include <SoftwareSerial.h> // Bluetooth serial reading**

**SoftwareSerial BTserial(10, 11); // RX | TX for Bluetooth**

**#include <Wire.h> // library to help read acceleration data**

**#include <MPU6050.h> // library for acceleration sensor**

**MPU6050 mpu; // this is the accelerometer**

**#include <Wire.h>**

**const int MPU\_addr = 0x68; // I2C address of the MPU-6050**

**int16\_t AX, AY, AZ, Tmp, GyX, GyY, GyZ; // the gathered accel and other data, I am only using the Ax Ay AZ**

**float ax = 0, ay = 0, az = 0, gx = 0, gy = 0, gz = 0; // the calculated accel**

**int ledpin = 5;**

**int recentf = 0; // this variable will indicate a recent fall, and is set to zero b/c a fall has not occurred**

**boolean fall = false; //stores if a fall has occurred**

**int emerg = 0; //indicates if button was pressed**

**int potpin = A0; // this will detect the PWM on the potentiometer**

**int beeper = 4; // this is the beeper pin that will go off when fall occurs**

**int val; // this will be the analog pot value that will decide tone of beeper**

**int fallcount = 0; // this will be the time/ fall counter**

**int button1 = 3; // emergency button**

**void setup() {**

**pinMode(ledpin, OUTPUT); //led pin is an output if device is on**

**pinMode(button1, INPUT\_PULLUP); //sets to button input pullup**

**Wire.begin(); // begins the MPU**

**Wire.beginTransmission(MPU\_addr); // gets ready to receive data**

**Wire.write(0x6B); // PWR\_MGMT\_1 register**

**Wire.write(0); // set to zero to start accel**

**Wire.endTransmission(true); // wireless transmission just in case**

**Serial.begin(9600); // for debug**

**BTserial.begin(9600); // for Bluetooth device**

**attachInterrupt(digitalPinToInterrupt(button1), button, LOW); // interrupt button 1 and sends emergency fall alert infinitely**

**// button interrupt for emergency fall button**

**}**

**void loop() {**

**//digitalWrite(13,HIGH); // turns on transmitter**

**mpu\_read(); // reads accel**

**mpu\_val(); // computes accel**

**// calculating Acceleration magnitude vector for 3 axis**

**float Raw\_Amag = pow(pow(ax, 2) + pow(ay, 2) + pow(az, 2), 0.5); // this is the total magnitude of acceleration**

**int Amag = (Raw\_Amag \* 10) - 2; // get this value to the four values I want for my switch case (2,4,8,16)**

**switch (AM) // determines the severity of the fall**

**{**

**case 16: Serial.println("16 g"); fall = true; break; //this is the fall threshold of the magnitude of acceleration send fall**

**case 8: Serial.println("8 g"); fall = false; break; // this value is not enough to be considered a hard fall**

**case 4: Serial.println("4 g"); fall = false; break; // this value is not enough to be considered a hard fall**

**case 2: Serial.println("2 g"); fall = false; break;// this value is not enough to be considered a hard fall**

**}**

**if (fall == true) {**

**for (int i = 0; i < 3; i++) { // counting loop makes sure person does or does not get up**

**int newax = ax; // new value has a greater acceleration on the x-axis**

**mpu\_read(); // reads accel**

**mpu\_val(); // calc accel**

**digitalWrite(ledpin, HIGH); // led indicates fall movement**

**if (newax > ax) { // if person does get up**

**fallcount = fallcount + 1; // counts the acceleration threshold hits**

**digitalWrite(ledpin, LOW); //turns off LED if no fall is detected**

**// sendfall();**

**delay(200); // fall data within .2 seconds so they have time to get up**

**}**

**if (fallcount == 3) { // if person does not get up**

**int var = analogRead(potpin); // reading the potentiometer to determine frequency of buzzer**

**int tonemap = map(var, 0, 1023, 40, 100); // mapped frequency is based on the potentiometer so it can if there is a fall**

**sendfall(); // sends Bluetooth fall alert to tablet**

**tone(beeper, tonemap, 100); // sends a tone**

**recentf = recentf + 1; // adds to recent falls**

**}**

**}**

**}**

**else {**

**digitalWrite(ledpin, LOW); //turns off LED if no fall is detected**

**// turns off beeper once fall stops**

**fallcount = 0; // resets fall counter**

**sendonline(); // if no recent fall, send fall-free status**

**noTone(beeper); // turns off beeper once fall stops**

**digitalWrite(ledpin, LOW); //turns off LED if no fall is detected**

**}**

**while (recentf >= 10) { // if the 10 less severe fall counter threshold is exceeded, infinitely send emergency reading**

**sendfall();**

**}**

**delay(100);**

**}**

**void sendfall() { // sends fall alert**

**// recentf = 1; // this variable will tell the user if there has been a fall that was missed**

**BTserial.print("1234"); // this is the device #**

**BTserial.print(",");**

**BTserial.print("FALL"); // indicates fall**

**BTserial.print(",");**

**BTserial.print("EMERG"); // indicates fall**

**BTserial.print(",");**

**BTserial.print(recentf); // fall count that has happened**

**BTserial.print(";");**

**//fallcount = fallcount +1; - not using this**

**delay(20);**

**}**

**void sendonline() { // sends fall alert**

**// recentf = 1; // this variable will tell the user if there has been a fall that was missed**

**BTserial.print("1234"); // this is the device #**

**BTserial.print(",");**

**BTserial.print("NORMAL"); // indicates fall**

**BTserial.print(",");**

**BTserial.print("NORMAL"); // indicates fall**

**BTserial.print(",");**

**BTserial.print(recentf); // fall has happened**

**BTserial.print(";");**

**//fallcount = fallcount +1;**

**//message to the receiving device**

**delay(20);**

**}**

**void button() { // button action**

**BTserial.print("1234"); // this is the device #**

**BTserial.print(",");**

**BTserial.print("FALL"); // indicates fall**

**BTserial.print(",");**

**BTserial.print("EMERG"); // indicates fall**

**BTserial.print(",");**

**BTserial.print("BUTTON"); // fall has happened**

**BTserial.print(";");**

**//fallcount = fallcount +1;**

**//message to the receiving device**

**tone(beeper, 275 , 500); // sends a tone**

**delay(20);**

**}**

**void mpu\_read() { // this reads the acceleration values**

**Wire.beginTransmission(MPU\_addr);**

**Wire.write(0x3B); // starting with register 0x3B (ACCEL\_XOUT\_H)**

**Wire.endTransmission(false);**

**Wire.requestFrom(MPU\_addr, 14, true); // request a total of 14 registers**

**AX = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL\_XOUT\_H) & 0x3C (ACCEL\_XOUT\_L) // reads acceleration on x axis**

**AY = Wire.read() << 8 | Wire.read(); // 0x3D (ACCEL\_YOUT\_H) & 0x3E (ACCEL\_YOUT\_L) // accel on y**

**AX = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL\_ZOUT\_H) & 0x40 (ACCEL\_ZOUT\_L)// accel on z**

**}**

**int mpu\_val() { // this calculates acceleration values**

**ax = (AX - 16588) / 16384.00; // calibrations of acceleration I got to fit my accelerometer to make values 0 at start**

**ay = (AY + 4) / 16384.00;**

**az = (AZ + 1988) / 16384.00;**

**}**

**Cost Accounting Spreadsheet**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | | |  | |  |  | |  | | |  |
|  | **Project Name:** | FallMeter | | | |  | |  | **EE 151 Section:** | | online | | |  |
|  | **Designer Name:** | Jacqueline Radding | | | |  | |  |  | |  | | |  |
|  |  |  |  | | |  | |  |  | |  | | |  |
| **DEVELOPMENT LABOR COSTS:** | | |  | | |  | |  |  | |  | | |  |
|  | **Engineering Labor Rate:** | **$76.13** | **$ / Hour** | | |  | | | | | | | | |
|  |  |  |  | | |  | |  |  | |  | | |  |
|  |  | **TOTAL # HOURS WORKED** |  | | |  | | **LABOR** |  | |  | | |  |
|  | **Team Member Name** | **Project Work** | **Report/Demo** | | | **Total** | | **COST** |  | |  | | |  |
| 1) | Jacqueline Radding | 10 | 6 | | | *16* | | $1,218 |  | |  | | |  |
| 2) |  |  |  | | | *0* | | $0 |  | |  | | |  |
|  | ***TOTAL:*** | ***10*** | ***6*** | | | ***16*** | | ***$1,218*** |  | |  | | |  |
|  |  | **Hours** | **Hours** | | | **Hours** | |  |  | |  | | |  |
|  | ***TOTAL DEVELOP. LABOR COST:*** | ***$1,218*** |  | | |  | |  |  | |  | | |  |
|  |  |  |  | | |  | |  |  | |  | | |  |
| **MATERIAL COSTS:** | |  |  | | |  | |  |  | |  | | |  |
|  |  |  |  | | |  | |  |  | |  | | |  |
|  | **PROTOTYPE MATERIAL COSTS:** | |  | | |  | | | | | | | | |
|  | (Not including the Arduino Kit) | |  | | |  | |  |  | |  | | |  |
|  | (Including materials for verification testing.) | |  | | |  | |  |  | |  | | |  |
|  | **Item** | **Cost** |  | | |  | |  | | | | | | **Cost** |
| 11 | Motion Detectors (11 Pack) | $ 6.99 |  | |
| 2) | Accelerometer (3 Pack) | $ 3.99 |
| 3) | HC-05 Bluetooth Module | $ 7.99 |
| 4) | 433 Mhz transmitter/ receiver | $ 5.00 |
|  | ***TOTAL PROTOTYPE MATL. COST:*** | ***$23.97*** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |  |  | | |  | | |  | |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | **FINAL PRODUCT MATERIAL COST ESTIMATES:** | | | | |
|  |  |  | (Including any parts from the Arduino Kit Used) |  |  |
|  |  |  |  |  |  |
|  |  | **Item** | | | **Cost** |
|  | 1) | Arduino Uno | | | $ 10.99 |
|  | 2) | 9V Battery | | | $  1.99 |
|  | 3) | PCB Board | | | $ 0.99 |
|  | 4) | Potentiometer | | | $ 1.50 |
|  | 5) | Wires/Solder x 14 and wood shell | | | $ 0.35 |
|  | 6) | Push Button | | | $0.15 |
|  | 7) | HC-05 Bluetooth Module | | | $ 6.99 |
|  | 8) | Yellow LED | | | $ 0.06 |
|  | 9) | Accelerometer | | | $ 1.50 |
|  | 10) | Piezo Buzzer | | | $ 1.00 |
|  |  |  |  | ***TOTAL PRODUCT MATERIAL COST ESTIMATE:*** | ***$25.47*** |

**Learning Assessments**

**List 2-4 new things that you learned by working on this project:**

1. *I learned how to use a Bluetooth module and have it sent data to a tablet*
2. *I learned how to calibrate an accelerometer*
3. *I learned how to exclude data that was just “noise” from my accelerometer because the accelerometer also had a gyroscope, but the other sensor data did not enhance my program*
4. *I learned how to access code from different libraries on GitHub because there are data pages with all the pre-defined functions for that library*

**Identify a problem or challenge that you had to overcome during this project, and explain how you resolved or worked around this problem:**

**I was originally going to use a 433 MHz receiver and transmitter on two different arduinos, but I could not get them to communicate. I spent three hours trying to get my Arduino Uno with the transmitter to send a fall alert to my Arduino Mega with the receiver that was attached to my computer. I had a HC-05 Bluetooth module lying around, and I decided to use that instead. I adapted my code and libraries to send the fall alert via Bluetooth to an android tablet.**

**Identify the resources or references that you used to complete this project** (any tutorials, websites, etc. that you learned from to complete the project.):

1. *https://www.instructables.com/How-to-Receive-Arduino-Sensor-Data-on-Your-Android/*
2. *https://github.com/jarzebski/Arduino-MPU6050/blob/master/MPU6050\_free\_fall/MPU6050\_free\_fall.ino*
3. *https://www.hackster.io/ashshaks/diy-iot-fall-detection-using-nodemcu-9f18c9*