Door Alarm

Strange 1984

Manpreet Singh, Sarangpreet Singh Padda, Harmanpreet Singh Ghotra

Computer Engineering 30 Section #04

Fall 2016

Preetpal Kang

December 21, 2016

Contents

Introduction

Project Description

Using the techniques, concepts and skills learnt in Cmpe30, a door alarm project has been designed which triggers beeping alarm when it detects motion in the door using the acceleration sensor on the SJ one board.

The primary SJ one board which act as door alarm is packaged into a box (See Fig #1). The SJ one board inside box is powered using 4 AA cells connected in series providing voltage of 6V to the board. The same voltage source powers the beeping alarm which is connected to the SJ one board. The accelerometer on the board is used to detect motion by calibrating its values. When the door alarm detects motion, it starts beeping alarm.

There is secondary SJ one board which act as remote control to shut off the alarm. The remote connects with the primary SJ one board using mesh network topology. When the main door alarm starts beeping, it can be shut off by pressing reset button on the remote control.

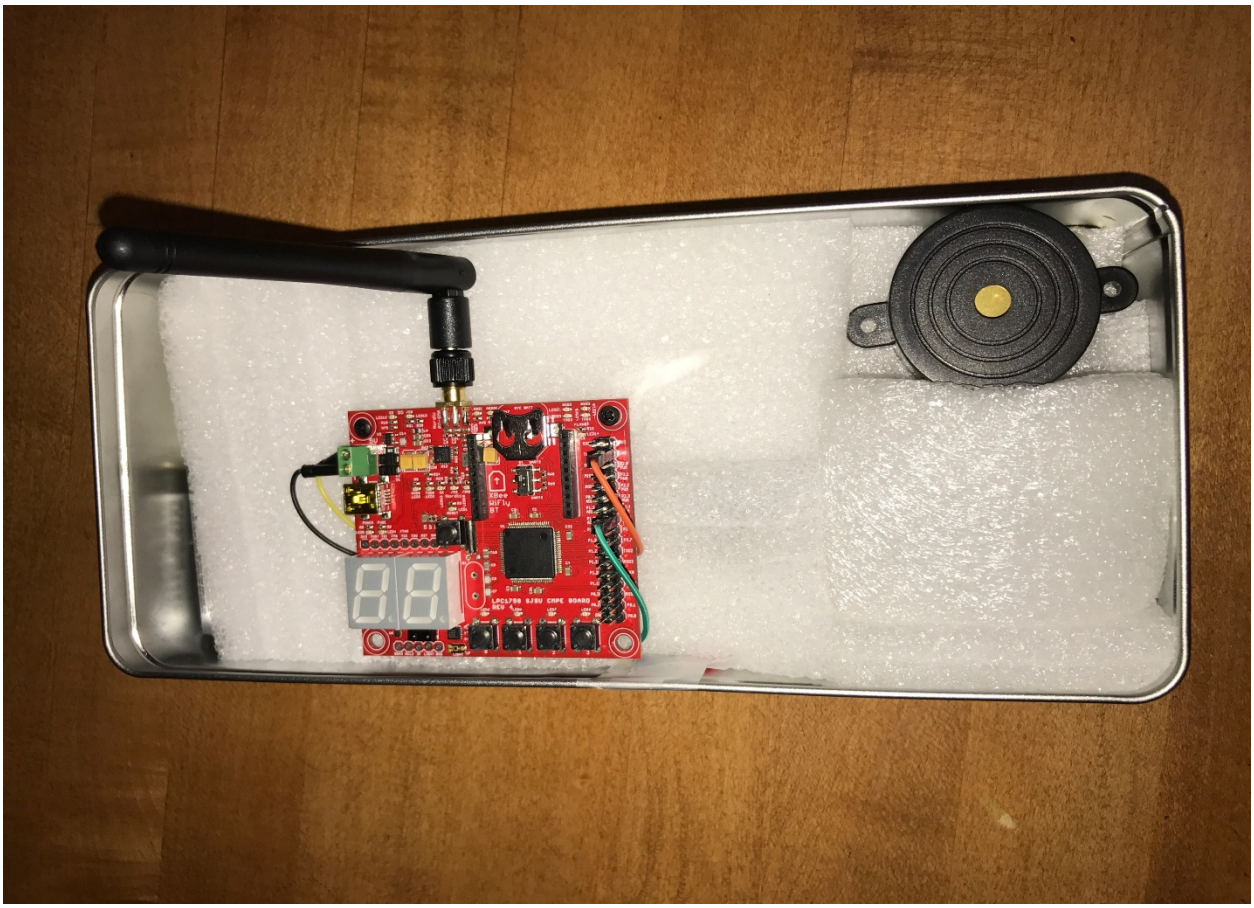

Fig #1- Door Alarm

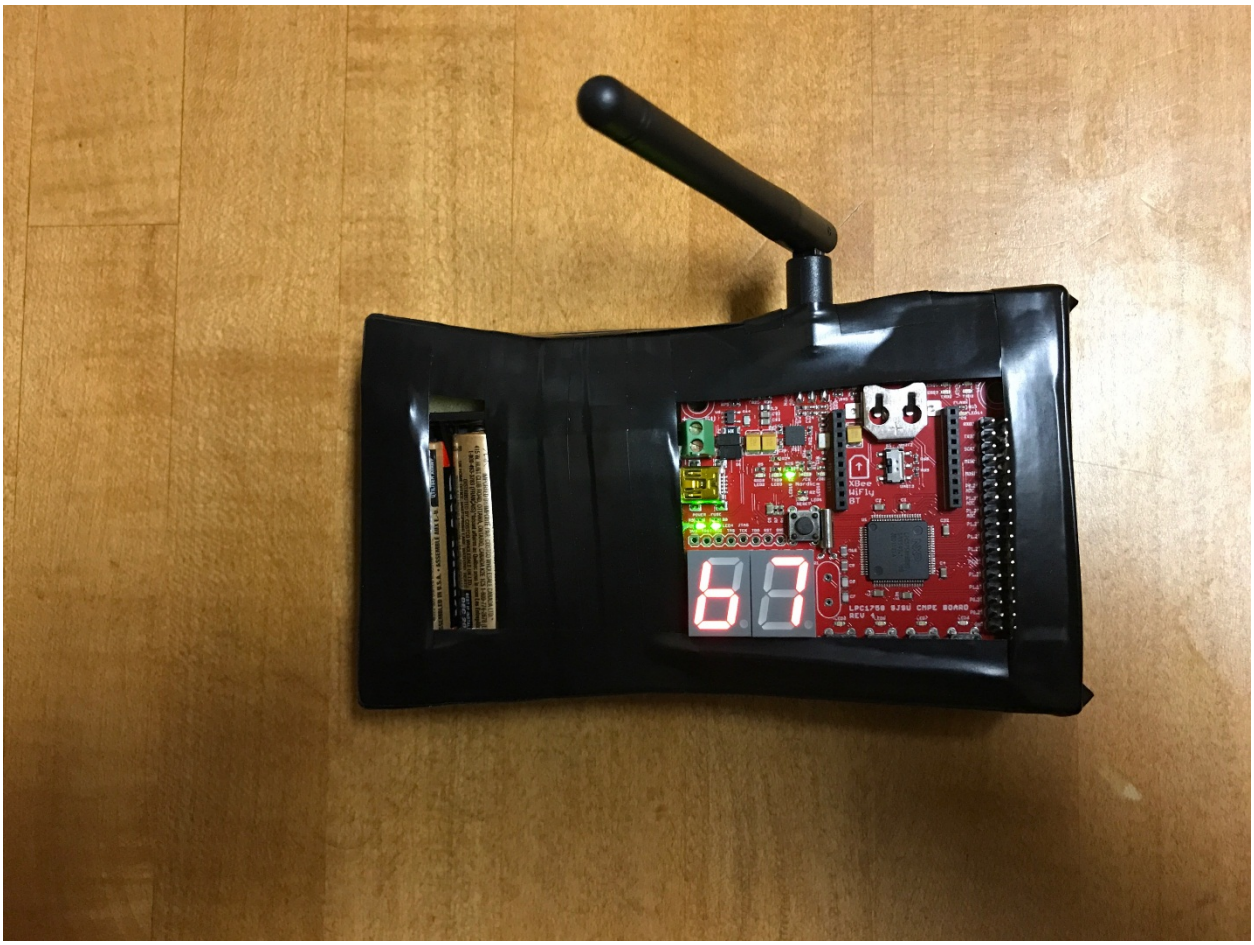

Fig #2 -Remote Control for shutting door alarm off

Features and Communication Table

Unique Features

The door alarm can be shut off wirelessly using the remote control which continuously sends packets to the door alarm to turn it off. Moreover, when the alarm is on from inside the door, a person can shut it off from outside using the remote control. After shutting it off, person can enter the door and again set the alarm on by pressing reset button the primary door alarm.

Team Roles and Responsibilities

Manpreet Singh

Built and Implemented the C code in the SJ one board and connected the two boards wirelessly using mesh network.

Sarangpreet Singh Padda

Worked on the hardware components of the project and ideology such as beeping alarm, jumper wires, mosfett etc. Also helped in the wiring connections of the project.

Harmanpreet Singh Ghotra

Worked on the packaging of the project and helped in the solving the issues of the packaging wires efficiently in the compact boxes.

Schematics

Parts List and Costs

Item	Details	Source	Cost / Item	Quantity	Total
Microcontroller	SJSU One Board with LPC1758	[1]	$80.00	x 2	$160.00
AA cell Holder	Connects the cells in series	Link	$1.00	x 2	$2.00
Alarm	Beeps when powered	Link	$2.50	x 1	$2.50
AA cells(1.5V)	AA cell/batteries for powering	Link	$1.00	x 8	$1.00
Antenna	Connects two boards wirelessly	Link	$6.00	x 2	$12.00
				Total	$177.50

The SJ one boards are bought from the professor Preetpal Kang. The AA cell holder, alarm and AA cells are bough from the store “Fries”. The two antennas are bought from Amazon online.

Schematic Layout

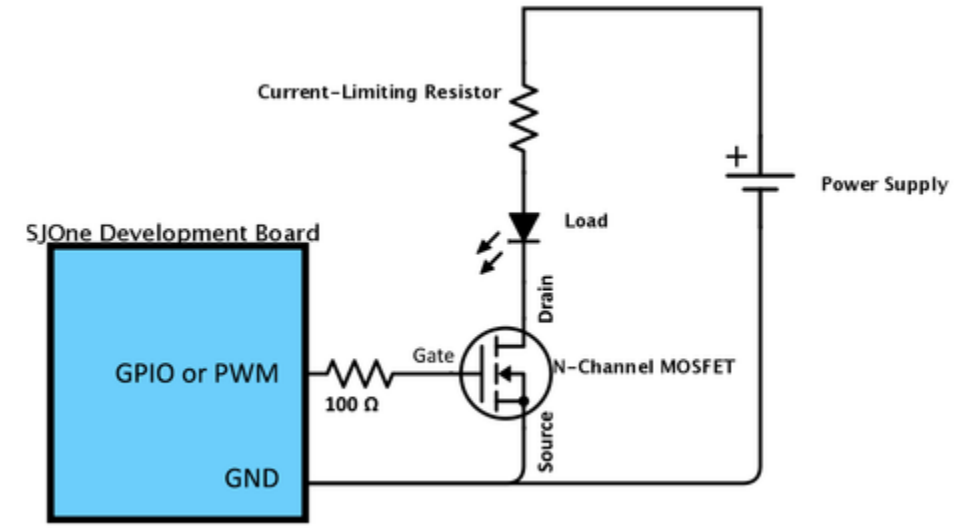

Fig #3- – Connections for MOSFET and SJ board

The above figure is the representation of the schematics of the SJ board, MOSFET, Load (beeping alarm), Power supply (4 AA cells in series) . The gate of the mosfet is connected with more than 5k resistor in our design which is connected with GPIO Pin 6 of the board. When is alarm is on (detection mode), the GPIO pin 6 is set low (0V). When the accelerometer detects motion, it sets GPIO pin 6 to high (5V) which trigger the beeping alarm.

The above described set up of the project is tested on the bread board before making final connections with the board.

Implementation

Flow Chart

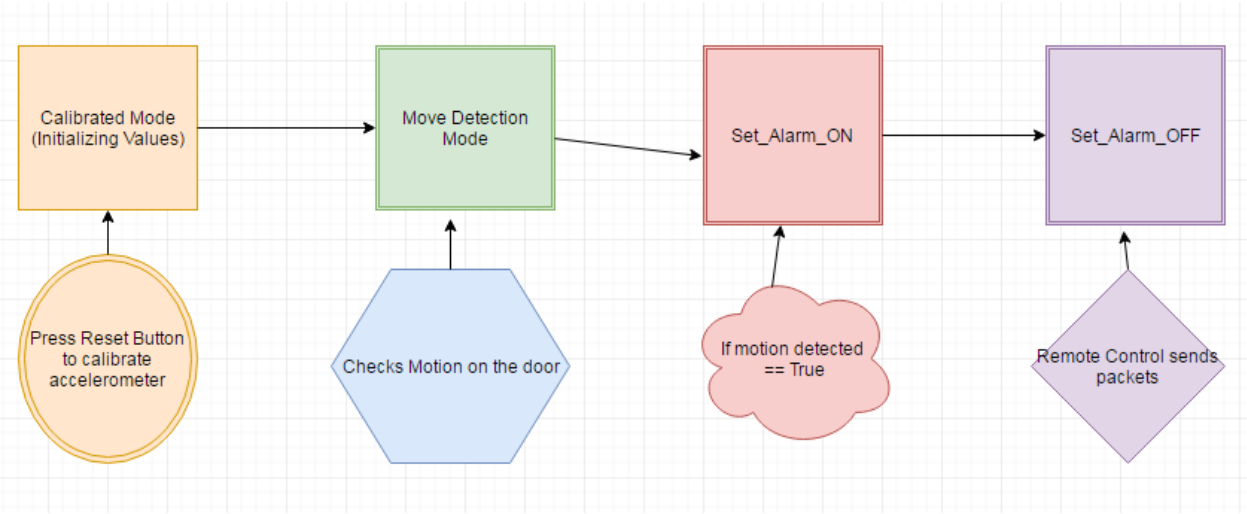

Fig #4- State Machine Layout for the Door Alarm

The state machine layout has four modes :- calibration mode, move detection mode, set_alarm_on mode, set_alarm _off mode. The calibration mode initializes the accelerometer values and set these values as standard values once they are set. After calibration mode, move detection mode gets on. In this mode, the accelerometer keeps on tracking the change in the values of its sensor. When there is deviation in the value of the sensor from the sensitivity value (variable tolerance set at 10), the set_alarm_on mode gets activated. In this mode, alarm keeps on beeping as the motion has been detected on the door. To shut off this alarm, the remote control board is powered on and reset button is pressed on it to continuously send packets to the door which set GPIO Pin 6 of the door board to low (0V) to shut off the alarm.

Description of Software

The C code has been designed and implemented for the functionality of the project. The in-built C functions has been used to enhance the performance and efficiency of the project such as `mesh_form_pkt(), mesh_send_formed_pkt()`, LE.on(1), delay_ms(100) etc.

The project is designed as a state machine so that it can switch states/modes as described in the flow char layout.

Code for Calibration Mode:-

*main.cpp | wireless.h | sys_co

```
853 void calibrateAccel(){
854  // reset alarm
855   moveDetected_temp =false;
856
857  //initialise x,y,z variables
858   xVal = AS.getX();
859   xMin = xVal;
860   xMax = xVal;
861
862   yVal = AS.getY();
863   yMin = yVal;
864   yMax = yVal;
865
866   zVal = AS.getZ();
867   zMin = zVal;
868   zMax = zVal;
869  for (int i=0; i<50; i++){
870  // Calibrate X Values
871  xVal = AS.getX();
872  if(xVal>xMax){
873  xMax=xVal;
874  }else if (xVal < xMin){
875  xMin=xVal;
876  }
877
878  // Calibrate Y Values
879  yVal = AS.getY();
880  if(yVal>yMax){
881  yMax=yVal;
882  }else if (yVal < yMin){
883  yMin=yVal;
884  }
885
886  // Calibrate Z Values
```

Fig #5- Function for Calibration

```
case calibrated:
    printf("i m in calibrated");

    calibrateAccel();
    LE.on(1);
    delay_ms(1000);
    printValues();
    calibrated_temp = true;
    current_state = moveDetected;
    LE.off(1);
    break;
```

Fig #6- Calibration Case Statement

The above code initializes the accelerometer sensor's values and set them as standard values when door alarm reset button is pressed.

Move Detection Mode:-

```
bool checkMotion(){
 bool tempB=false;

 xVal = AS.getX();
 yVal = AS.getY();
 zVal = AS.getZ();

 if(xVal >(xMax+tolerance)||xVal < (xMin-tolerance)){
 tempB=true;
 printf("X Failed = ");
 printf("%i\n\n",xVal);
 }

 if(yVal >(yMax+tolerance)||yVal < (yMin-tolerance)){
 tempB=true;
 printf("Y Failed = ");
  printf("%i\n\n",yVal);
 }

 if(zVal >(zMax+tolerance)||zVal < (zMin-tolerance)){
 tempB=true;
 printf("Z Failed = ");
  printf("%i\n\n",zVal);
 }

 return tempB;
}


 case moveDetected :
     printf("i m in move detecttedd");

     if(calibrated_temp){
         if(checkMotion()){
             moveDetected_temp = true;
             current_state = set_alarm_on;

         }
     }//ends moveDetected - if statements bracket
     break;
```

Fig #7- Function for Detecting Motion

Alarm ON Mode:-

```
case set_alarm_on :
    printf("i m in alarm on");

    LE.on(2);
    LE.on(4);
    delay_ms(100);
    LE.off(2);
    LE.off(4);
    delay_ms(100);
     myPin.setHigh();      // Pi
    delay_ms(100);
    break;
```

Alarm OFF Mode:-

```
case set_alarm_off:
    mesh_packet_t pkt1;
    addr = 106;
    hops = 10;
    alarm_off =true;
    printf("alarm off.... you got it...hacker");
    LE.on(3);
    delay_ms(100);
    LE.off(3);
    delay_ms(100);
    myPin.setLow();

    mesh_form_pkt(&pkt1, addr, mesh_pkt_ack, hops,
    1,                      /* 2 Pairs below */
    &alarm_off, sizeof(alarm_off)   /* Pair 1 */
    );  /* Pair 2 */

     mesh_send_formed_pkt(&pkt1);

    break;
```

Fig #8 Alarm OFF & Alarm ON Function

Analyzing the Code

The C code has been designed using switch/case statements for the transitioning of the four modes in the project. The calibratedAccel() function initializes the value of the accelerometer sensor. The checkMotion() function checks the motion or detects the change in the accelerometer sensor's values.

Testing

Tests Performed

In order to test the project, Double sided tape is used to fix the door alarm box on the door. Then, the door is opened from outside to check how much motion is needed to trigger alarm. This helped in setting the variable tolerance a fixed value which is needed for the sensitivity of the door alarm. Moreover, the Hercules terminal is used to print the values of accelerometer sensor. The remaining testing is done on the breadboard to test the wire connections.

Problems Encountered

The project worked as expected. The only problem faced was delay in the transmitting signal form remote control to main door alarm because the remote control was continuously sending signals to the main door alarm and main door alarm was expected to catch the signal immediately and shut off the alarm. But the alarm was shutting off after 2-3 minutes seconds of delay which should not happen.

Solutions and Debugging

The delay problem of the signal eas solve by using the antennas on the both main door alarm SJ one board and in the remote control. The two antennas transmit and receive signals very fast and amplify the signals. By connecting antennas with SJ one board, the delay of time was shortened from 2-3 minutes to 20-25 seconds of delay.

Conclusion

What we Learned

Through this class and project, we learned great coding skills, ideologies for making real life projects. We learned how to work in the project along with other teammates. This project exposed us to vast growing possibilities of creative applications an computer engineer can work on.

What Worked and Didn't Work

The project worked as expected perfectly. No doubt, we faced some challenges and issues in the working of the design, but overall, the hard work and patience helped us in aheiving our aim of the project.

Improvements

There are always possibilities to improve anything. Similarly, this project can be improved by a lot of factors. The main improvement that can be made is more compact packaging of the door alarm box. It should made more small and compact so that it looks better to the user.

Acknowledgements

We would like to acknowledge Professor Preetpal Kang and her TA Sara and all the ISA team members.

Code

```
void calibrateAccel();
bool checkMotion();
int tolerance=10; // Sensitivity of the Alarm
//bool calibrated=false; // When accelerometer is calibrated - changes to true
//bool moveDetected=false; // When motion is detected - changes to true
bool calibrated_temp =false;
bool moveDetected_temp =false;

int xMin; //Minimum x Value
int xMax; //Maximum x Value
int xVal; //Current x Value

int yMin; //Minimum y Value
int yMax; //Maximum y Value
int yVal; //Current y Value

int zMin; //Minimum z Value
int zMax; //Maximum z Value
int zVal; //Current z Value

int main(void)

{

	//alarmcode
	int temperature_f = TS.getFarenheit();
	LD.setNumber(temperature_f);
	typedef enum { calibrated, moveDetected,set_alarm_on, set_alarm_off }
motion_var ;
                                motion_var current_state;

                                GPIO myPin(P2_5);   // Control P1.19
                                 myPin.setAsOutput(); // Use the pin as output pin
                                char  addr ;
                                char hops;
                                bool alarm_off ;

                                mesh_packet_t pkt;
                                int count =0; //glob var
                                while(1){
                                       delay_ms(100);
                                       bool sw1 =SW.getSwitch(1);
                                              if(sw1){
                                                     current_state =calibrated ;
                                              }
                                       bool sw2 =false;
                                       delay_ms(100);


                                                                    // background task takes
care of the mest network
```

```
        /* Try to get a packet destined for us with 100ms timeout*/

            if (wireless_get_rx_pkt(&pkt, 100)) {

               mesh_deform_pkt(&pkt, 1, &sw2, sizeof(sw2) );

               printf("data %i\n\n",pkt.data[0]);

               printf("switch 2 value received as %i\n", sw2);


            }
                                        if (sw2==1){
                                               count =count +1;

                                        }
                                        if(count>=1){
                                               current_state = set_alarm_off;
                                        }
                                        switch(current_state)
                                        {
                                        case calibrated:
                                               printf("i m in calibrated");

                                               calibrateAccel();
                                               LE.on(1);
                                               delay_ms(1000);
                                               printValues();
                                               calibrated_temp = true;
                                               current_state = moveDetected;
                                               LE.off(1);
                                               break;

                                        case moveDetected :
                                               printf("i m in move detecttedd");

                                               if(calibrated_temp){
                                                      if(checkMotion()){
                                                             moveDetected_temp = true;
                                                             current_state =
set_alarm_on;


                                                      }
                                               }//ends moveDetected - if statements
bracket
                                               break;

                                        case set_alarm_on :
                                               printf("i m in alarm on");

                                               LE.on(2);
                                               LE.on(4);
                                               delay_ms(100);
```

```
                                        LE.off(2);
                                        LE.off(4);
                                        delay_ms(100);
                                         myPin.setHigh();     // Pin will now
be at 3.3v
                                        delay_ms(100);
                                        break;

                                    case set_alarm_off:
                                        mesh_packet_t pkt1;
                                        addr = 106;
                                        hops = 10;
                                        alarm_off =true;
                                        printf("alarm off.... you got
it...hacker");
                                        LE.on(3);
                                        delay_ms(100);
                                        LE.off(3);
                                        delay_ms(100);
                                        myPin.setLow();

                                        mesh_form_pkt(&pkt1, addr,
mesh_pkt_ack, hops,
                                  1,                    /* 2 Pairs below */
                                   &alarm_off, sizeof(alarm_off)   /* Pair 1 */
                                     );  /* Pair 2 */

                                         mesh_send_formed_pkt(&pkt1);

                                        break;

                                    default:
                                        printf("check your code fool");
                                    }

                                }



                                return 0;


                            }


void calibrateAccel(){
 // reset alarm
  moveDetected_temp =false;

 //initialise x,y,z variables
  xVal = AS.getX();
  xMin = xVal;
  xMax = xVal;
```

```
  yVal = AS.getY();
  yMin = yVal;
  yMax = yVal;

  zVal = AS.getZ();
  zMin = zVal;
  zMax = zVal;
 for (int i=0; i<50; i++){
 // Calibrate X Values
 xVal = AS.getX();
 if(xVal>xMax){
 xMax=xVal;
 }else if (xVal < xMin){
 xMin=xVal;
 }

 // Calibrate Y Values
 yVal = AS.getY();
 if(yVal>yMax){
 yMax=yVal;
 }else if (yVal < yMin){
 yMin=yVal;
 }

 // Calibrate Z Values
 zVal = AS.getZ();
 if(zVal>zMax){
 zMax=zVal;
 }else if (zVal < zMin){
 zMin=zVal;
 }

 //Delay 10msec between readings
 delay_ms(100);
 }
 calibrated_temp =true;

}



//Function used to detect motion. Tolerance variable adjusts the sensitivity of
movement detected.
bool checkMotion(){
 bool tempB=false;

 xVal = AS.getX();
 yVal = AS.getY();
 zVal = AS.getZ();

 if(xVal >(xMax+tolerance)||xVal < (xMin-tolerance)){
 tempB=true;
 printf("X Failed = ");
 printf("%i\n\n",xVal);
 }
```

```
 if(yVal >(yMax+tolerance)||yVal < (yMin-tolerance)){
 tempB=true;
 printf("Y Failed = ");
  printf("%i\n\n",yVal);
 }

 if(zVal >(zMax+tolerance)||zVal < (zMin-tolerance)){
 tempB=true;
 printf("Z Failed = ");
  printf("%i\n\n",zVal);
 }

 return tempB;
}
```