Martin Solis and Alonzo Marsh
ECE 110 Honors Lab
December 14th, 2015

ECE 110 Honors Final Report

The Smart Safety Hat

INTRODUCTION

Problem Description

According to the Occupational Safety and Health Administration (OSHA), the four most frequent accidents that resulted in the loss of life are falls (39.5% of fatalities), electrocution (8.5%), struck by an object (8.4%), and caught inbetween (1.4%). The most common hazards that construction works face daily also include extreme temperature, fire, and noxious gases. In all of these cases, it is vital for workers to be able to notify their managers as soon as possible so the appropriate assistance can be provided.

Proposed Solution

Create a hard hat that provides greater security for the wearer and others around them by incorporating sensors that wirelessly stream data to a central server. This allows for threat assessment around the workplace and improves response times for on site emergencies. Also allows for emergency communications between supervisors and employees anywhere on the complex.

Moreover, by including sensors that measure various aspects of the wearer's surroundings, the hat can accurately determine whether or not there is a danger to the wearer and alert the wearer. To prevent false alarms, there are also options for the user to cancel alarms when the hat throws a false alarm.

Design Description

Block Diagram

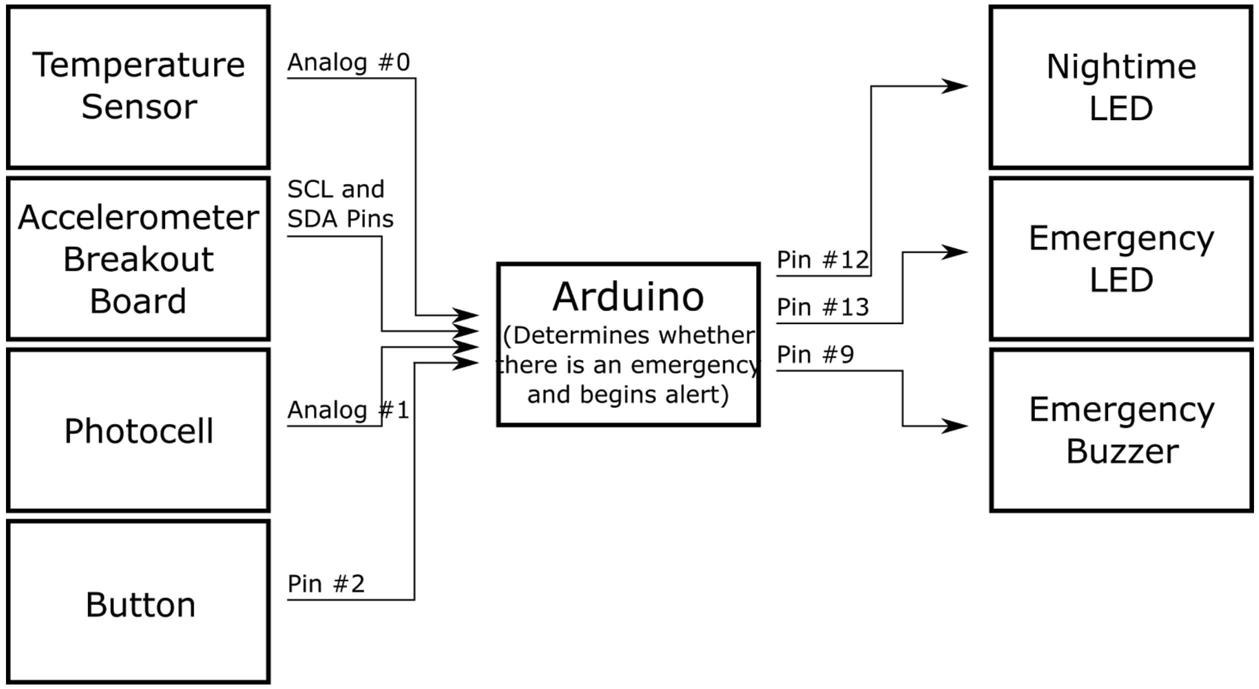


Figure 1: Block diagram showing the 4 inputs into the arduino and the three outputs.

The Arduino takes input from for different sensors through 2 analog pins and 3 digital pins (2 of which are the SCL and SDA pins). The arduino then processes the inputs to determine whether or not the values from the pins exceed safety standards. Then determines whether the nighttime light needs to blink or, in the event of emergency, sounds the emergency buzzer, blinks the emergency light and turns the nighttime light on (not blinking).

Smart Hat Pictures

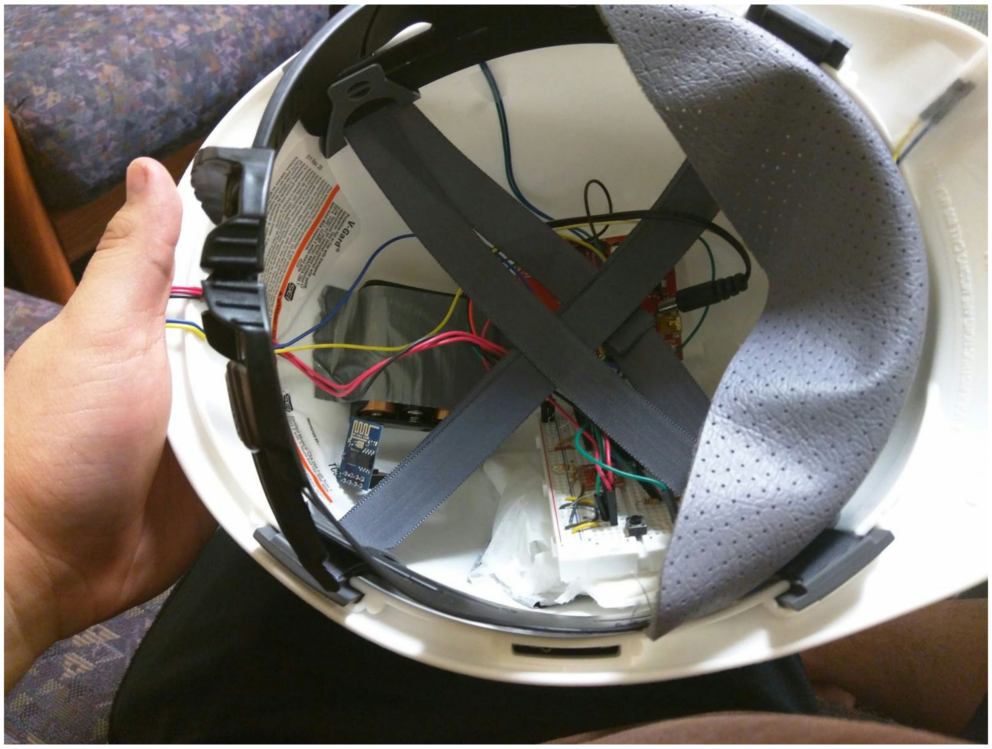


Figure 2: Block diagram showing the 4 inputs into the arduino and the three outputs.

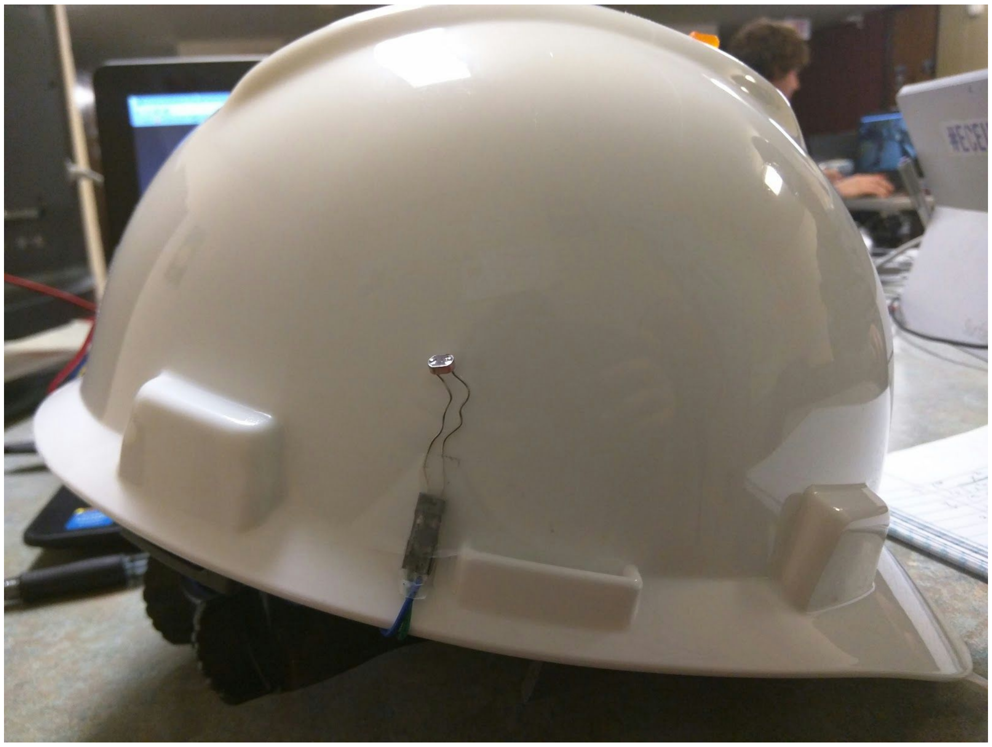


Figure 3: Block diagram showing the 4 inputs into the arduino and the three outputs.

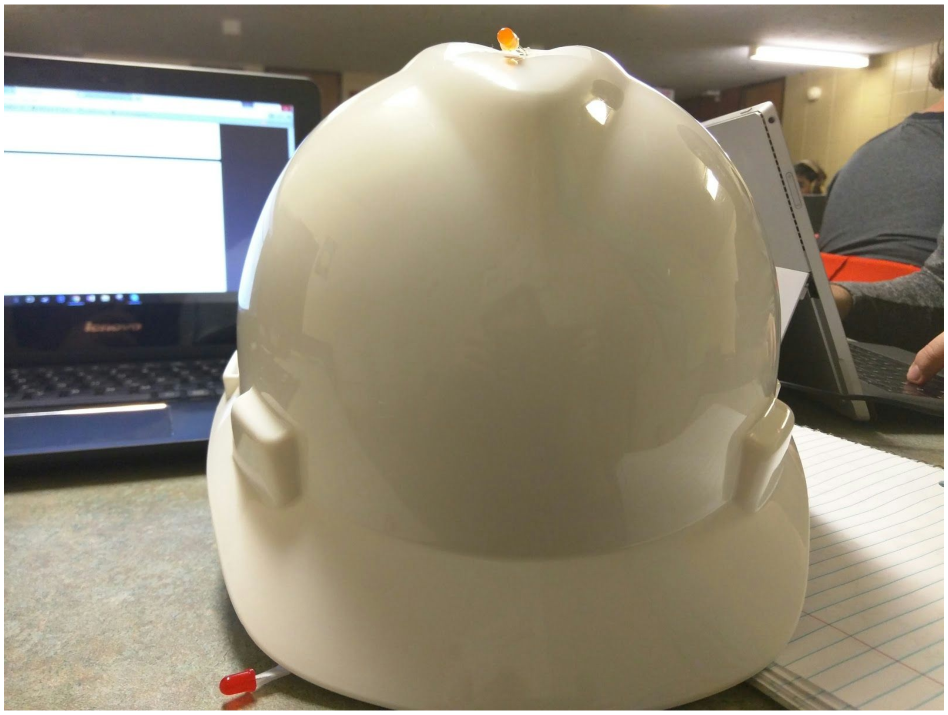
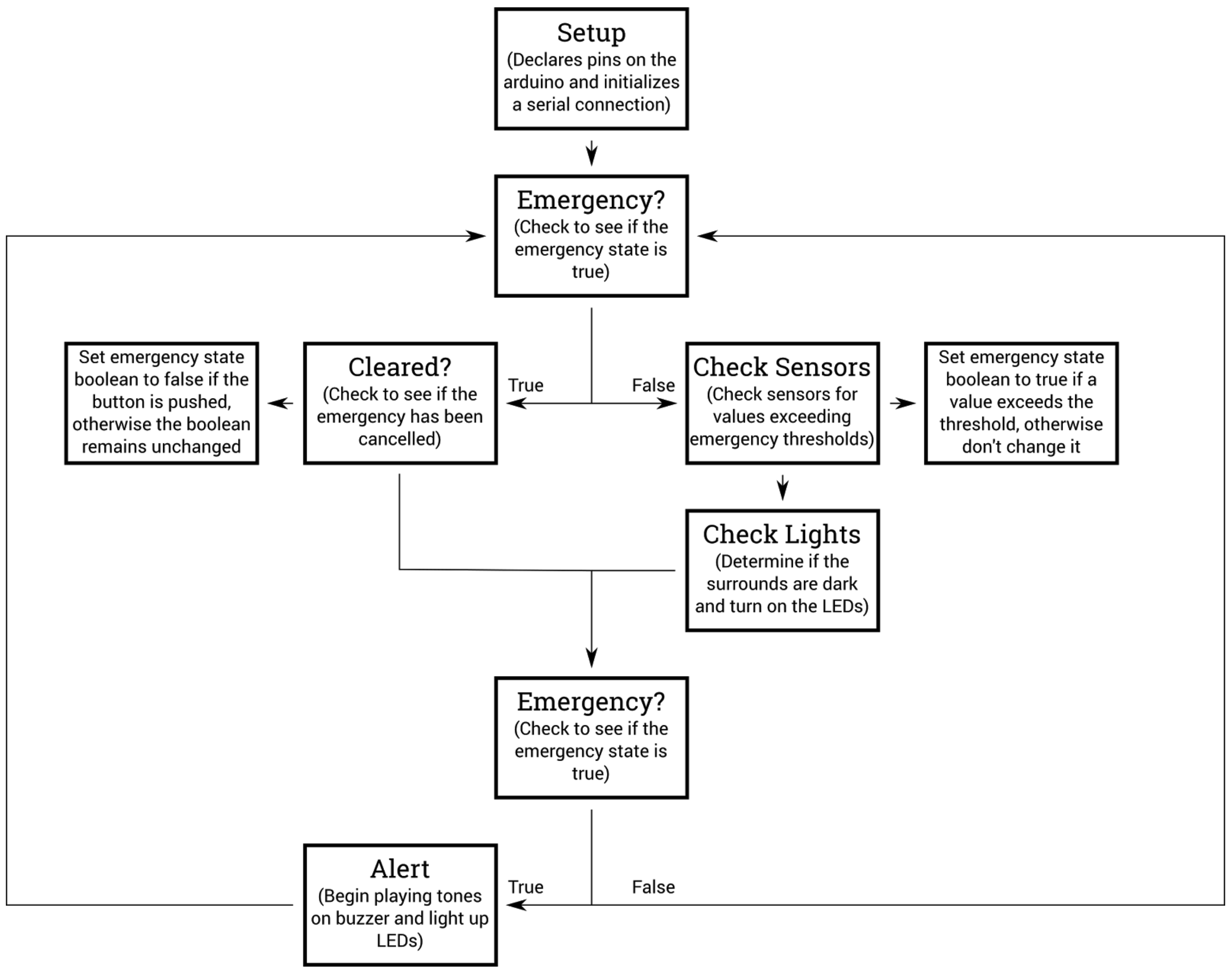


Figure 4: Block diagram showing the 4 inputs into the arduino and the three outputs.

Software Flow Chart



Setup
(Declares pins on the arduino and initializes a serial connection)
Emergency?
(Check to see if the emergency state is true)
True
False
Set emergency state boolean to false if the button is pushed, otherwise the boolean remains unchanged
Cleared?
(Check to see if the emergency has been cancelled)
Check Sensors
(Check sensors for values exceeding emergency thresholds)
Set emergency state boolean to true if a value exceeds the threshold, otherwise don't change it
Check Lights
(Determine if the surrounds are dark and turn on the LEDs)
Emergency?
(Check to see if the emergency state is true)
Alert
(Begin playing tones on buzzer and light up LEDs)
True
False

Figure 5: Code flow chart outlining the main processes of the arduino code loop.

Arduino Code

```
#include <SFE_MMA8452Q.h>
#include <Wire.h>

int loopCount = 0;
const int emergencyLEDPin = 13; //The pin that the LED is connected to the arduino across.
//String emergencyType = "null";
//String emergencyLog = "";
const int nightTimeLEDPin = 12;
const int lightSensorPin = 1;
const int lightLevelCutoff = 175;
boolean emergencyState = false;
const int buzzerPin = 9;
const int songLength = 4;
char notes[] = "cgcg";
int beats[] = {2, 2, 2, 2};
int tempo = 150;
```

```
const int button1Pin = 2;  // pushbutton pin
const int temperaturePin = 0; //temperature sensor pin
const int cutoffTemperature = 120; //degrees F
MMA8452Q accel;
const float freeFallCutoff = 0.2;

void setup() {
  Serial.begin(9600);
  Serial.println("1");
  pinMode(emergencyLEDPin, OUTPUT);
  Serial.println("2");
  pinMode(nightTimeLEDPin, OUTPUT);
  Serial.println("3");
  pinMode(buzzerPin, OUTPUT);
  Serial.println("4");
  pinMode(button1Pin, INPUT);
  Serial.println("5");
  accel.init();
  Serial.println("6");
  Serial.println("Initialized and Ready to Go");
}

void loop() {
  Serial.println(loopCount);
//  Serial.print("Emergency = ");
//  Serial.println(emergencyType);
  if (emergencyState) {
    emergencyState = !checkButton();
    checkForEmergency();
  } else {
    emergencyState = checkForEmergency();
    setupLights();
  }
  if (emergencyState) {
    alert();
  }
  loopCount++;
  //  delay(1000);
}

boolean checkForEmergency() {
  if (checkTemperature(cutoffTemperature)) {
//    emergencyType = "Temperature Alert";
//    emergencyLog = emergencyLog + loopCount + " " + emergencyType + "\n";
    return true;
  }
  if (readAccelerometerMagnitude() < freeFallCutoff) {
//    emergencyType = "Accelerometer Alert";
//    emergencyLog = emergencyLog + loopCount + " " + emergencyType + "\n";
    return true;
  }
  return false;
}

boolean checkButton() {
  int button1State;  // variables to hold the pushbutton states
  button1State = digitalRead(button1Pin);

  if (button1State == LOW) {
    Serial.println("Button Pushed");
    return true;
  }
  Serial.println("Button Not Pushed");
  return false;
}

boolean checkTemperature(int cutoff) {
  float voltage, degreesC, degreesF;
  voltage = getVoltage(temperaturePin);
  degreesC = (voltage - 0.5) * 100.0;
  degreesF = degreesC * (9.0 / 5.0) + 32;
  Serial.print("voltage: ");
  Serial.print(voltage);
  Serial.print(" deg C: ");
  Serial.print(degreesC);
  Serial.print(" deg F: ");
  Serial.println(degreesF);
  return degreesF >= cutoff;
}

void alert() {
  //Light Up Emergency LED
  digitalWrite(emergencyLEDPin, HIGH);
  digitalWrite(nightTimeLEDPin, HIGH);
  //play Alert Tone
  int i, duration;

  for (i = 0; i < songLength; i++)
  {
    if (i == songLength / 2) {
      digitalWrite(emergencyLEDPin, LOW);
    }
    duration = beats[i] * tempo;
    if (notes[i] == ' ')
    {
      delay(duration);
    }
    else
    {
      tone(buzzerPin, frequency(notes[i]), duration);
      delay(duration);
    }
    delay(tempo / 10);
  }
}

int frequency(char note)
{
  int i;
  const int numNotes = 8;
  char names[] = {'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C'};
  int frequencies[] = {262, 294, 330, 349, 392, 440, 494, 523};
```

```
 for (i = 0; i < numNotes; i++)
 {
   if (names[i] == note)
   {
     return (frequencies[i]);
   }
 }
 return (0);
}

float getVoltage(int pin)   {
 return (analogRead(pin) * 0.004882814);
}

void setupLights() {
 Serial.println("Reading");
 double lightLevel = analogRead(lightSensorPin);
 Serial.print("Light Level = ");
 Serial.println(lightLevel);
 if (lightLevel < lightLevelCutoff && loopCount % 14 < 7) {
   digitalWrite(nightTimeLEDPin, HIGH);
 } else {
   digitalWrite(nightTimeLEDPin, LOW);
 }
}

float readAccelerometerMagnitude() {
 if (accel.available())
 {
   accel.read();
 }
 float x, y, z;
 x = accel.cx;
 y = accel.cy;
 z = accel.cz;
 float mag = sqrt(x * x + y * y + z * z);
 Serial.print("Magnitude of Accel = ");
 Serial.println(mag);
 return mag;
}
```

Results

Overview of Results

The hat worked as expected according to the original design. While there were some serious sensitivity issues, the average response times for most situations was as we expected them to be.

Qualitative Analysis

While we were testing our device, we noticed various issues with the cutoffs that we set for the arduino code. Namely, that the temperature sensor cutoff was too low, the accelerometer's check period was too short, and the photoresistor was actually pretty good. The temperature sensor, partly due to it's design, proved to be very slow to detect an emergency because the device itself has to heat up. There were also issues with clearing the alarm because the device took roughly ten seconds to cool down. The accelerometer was the most common cause of false alarms because it only checked at one point in the cycle. While the check level for the photoresistor was pretty accurate, we did notice that the light had a tendency to flicker intermittently around the checkpoint. Finally, we would have benefitted from a louder buzzer. While it worked well for attracting the attention of the wearer in labs, we had trouble hearing it while testing it in public locations, which means it would likely be unfit for use in a construction area.

Quantitative Analysis

The average response time for fire alerts was 3 seconds. This was likely due to the relatively slow response rate of the temperature sensor. Specifically, it took quite a while to heat up and cool down, resulting in longer response times. The average response time for falls was less than half a second, which in many cases during testing proved to be very problematic because simply running quickly or walking down stairs could result in triggering emergency alarms. Finally, the average response for the nighttime lights was less than a second, which was roughly where we wanted it.

Future Work

Next Steps for Project

As we approach Engineering Open House and continue to develop the helmet, we plan on adding more sensors. The sensors we would like to add the most are vibration to detect objects hitting the head, gps to determine location, and CO_2 to detect dangerous gases. We plan on making the WiFi shield work so that it will be able to relay logs to a base of operations of the conditions around the user. We would also like to improve on the current sensors. Currently, the accelerometer has a problem of setting off the emergency sequence when walking and/or running on a flight of stairs. The temperature sensor will continue triggering the emergency sequence until it has cooled down, which is an issue because the button to deactivate the emergency sequence is rendered useless until it sense temperature below the temperature threshold. Since electric shock is the second most cause of death, we would like to add on some sort of metal connection from the brace that connects the head to the battery to redirect the shock into the battery and short circuit the helmet as opposed to shocking the wearer.

Conclusion

What Worked

The two sensors that were implemented worked as intended, accelerometer and temperature sensor. The accelerometer will constantly read one G of acceleration going through it. Once the helmet enters free fall, it will read 0Gs as it is accelerating at about 9.81 m/s^2. The code is set so that once the accelerometer reads below 0.2Gs, it will activate the emergency sequence. The temperature sensor exceed the cutoff when it would reach temperatures above 130 degrees Fahrenheit and would then ring the alarm for an extended period of time until it had cooled down. We also attached a Wifi antenna, however we did not have a server for it to output its data to, so we were unable to implement this.

Potential Improvements

As stated earlier, we would like to continue to add more sensors to test for other serious causes for injury and/or death. We would like to cut down on the weight and combine the arduino and breadboard into one pcb integrated into the helmet as well as implement a rechargeable power supply. The next step after adding more sensors is to create a base that the helmet (possible multiple helmets) would be able to send logs of any emergency conditions where someone would be able to contact everyone in the event of the emergency and have the situation taken care of.

Lessons Learned

There are many different causes of accidents on construction sites, sometimes resulting in debilitating injuries and death. The potential benefits of a network that workers can use to notify each other of injuries increases the potential for the damages from accidents to be minimized, and in some cases avoided completely.

While working on this project, we were able to better understand the dangers that workers face, and how it is possible to detect when danger is imminent. We also learned how to take a theoretical design and bring it to the prototyping stage.