Electromagnetic Backpack Lock (Taser)

- Diego Esquivel (diegoe2, ECE 120)
- Fred Chang (kehangc2, ECE 110)
- Alec Saebeler (aws2, ECE 110)

Introduction

There are over 83 millions students, in the US alone, and 79 million of those wear backpacks. The reliance on backpacks by students is obviously tremendous. Unfortunately, theft is far more common than any other crime on college campuses. We want to lower the amount of thefts that occurs with students so we don't all lose our very expensive belongings because if there is one thing students can not live without, it would be the backpack. The backpack: a convenient vessel for school work, textbooks, electronics, food (most important), and other miscellaneous yet necessary items. We proposed a new type of lock, one that both secures the backpack and provides an passive system to notify the owner. Our design would incorporate a Hall effect sensor paired with a magnet. This sensor system would connect to an Arduino logic board which processes the signal and indicates whether to sound the alarm or unlock for the user.

Design

As far as design go the only real decision, that resulted from any sensors having to work with other pieces in a relatively small area, was to utilize a regular magnet, as opposed to an electromagnet. The reason for this was that we couldn’t find an electromagnet that was small enough to fit into our design. Making this decision did take us a step back, in that we couldn’t

just ultimately cut power to the alarm or the “lock” unlock. Instead we were limited to only being able to control whether the alarm went on or off, seeing as we can’t make a normal magnet from fulfilling its sole purpose in life, to be an electromagnet. This change of plans altered our logic circuit, which we handled by inverting our signal and adding code for the Arduino to process. The Arduino was still used to manage the numeric pad. This method of user input would allow us to have a four-digit code we could use to ‘unlock’ the circuit.

Parts & Materials List

Arduino

Hall effect sensor

Logic Gates (AND,OR,NOR, etc)

LED(s)

Electromagnet (Small)

Numeric Pad, voice recognition shield, or combo lock

Resistors

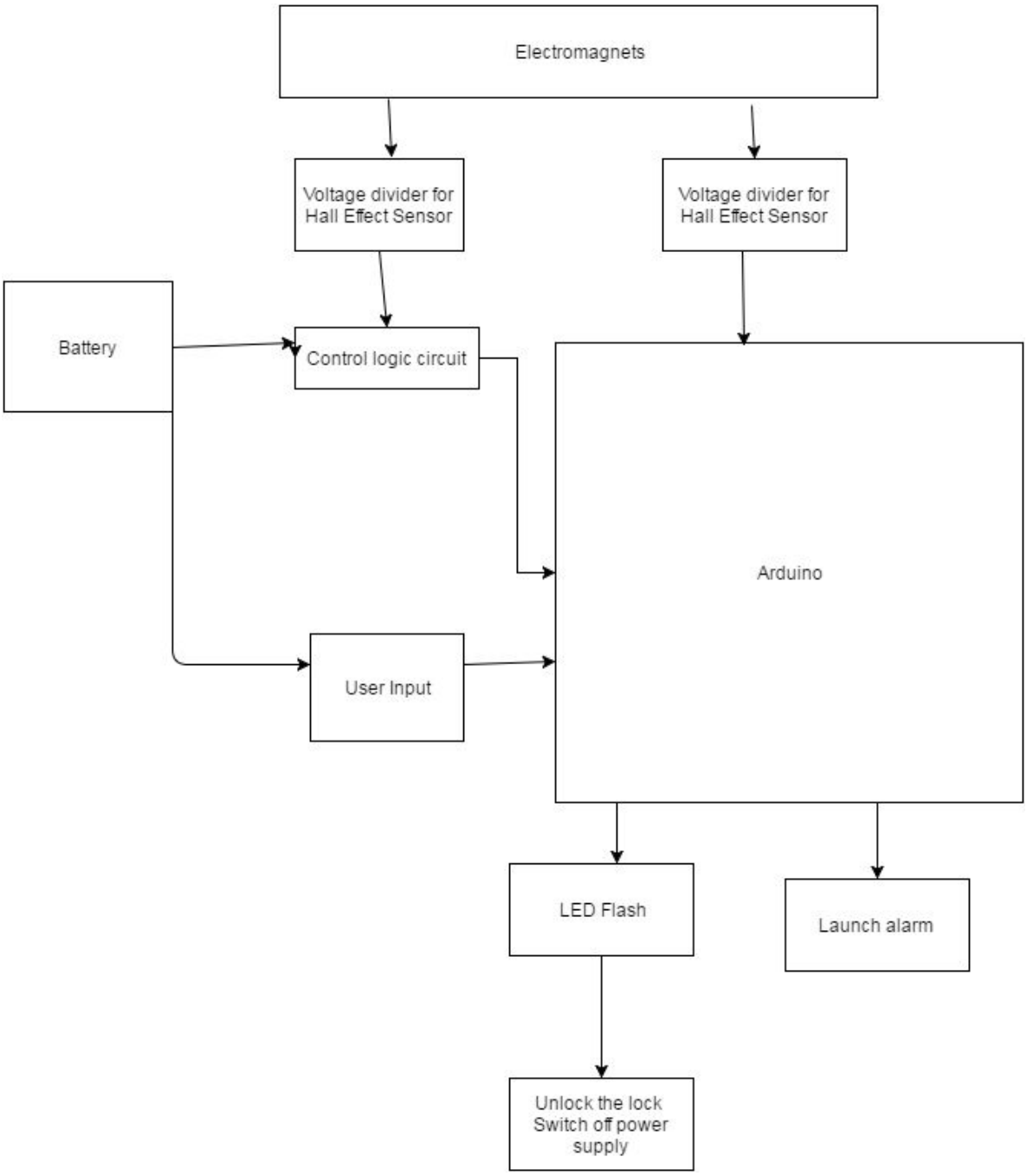
Wires

Backpack

Buzzer/Alarm

Breadboard

Block Diagram



Conclusion

Our circuit performed well as a proof of concept, but lacks many of the bells and whistles of the original design. We used an LED instead of a buzzer, did not build/test the system with a backpack, and were unable to procure and wire a functioning physical lock. The logic states worked well, the sensor worked well, we just did not have the time to build a working prototype with a locking mechanism.

One of the initial obstacles that was encountered in the design process was that our input/output signals would vary because the voltage that was being delivered to any sensors and gates did not jump from 5V to 1V instantaneously, rather it would move relatively slowly from 5V to 1V. Fixing this, fortunately, was not too big of an obstacle. All we had to do was invert the voltage twice by passing the voltage through a NOT gate, thereby making 5V a 0, 0V a 1, and anything in between wouldn't be read or sent as a signal to the hall effect sensor. This, however, was not the input we wanted. We actually wanted the exact opposite so we simply just passed the output through the NOT gate one more time and then that output got sent to the hall effect sensor.

While testing our circuit, to ensure that it worked correctly, we noticed that we were getting states that we weren't supposed to be getting, such as the alarm sounding when the magnet was still next to the hall effect sensor. At first we were confused as to why this was occurring when moments before it was working perfect. By checking the output of the hall effect sensor, through the use of a voltmeter, we quickly noticed that our hall effect sensor had actually stopped working. Fortunately for us we had ordered plenty of hall effect sensors in anticipation of this possibly happening. We made attempts at trying to prevent this from happening again, but

this only resulted in the destruction of even more hall effect sensors, and so, with a limited supply, we decided to cease attempts at fixing this issue to ensure we had at least one sensor working to present with.

From the very beginning, the one issue we knew was going to be inevitable was coding in the Arduino language. With none of us having any prior knowledge of the language itself this had to have been the most difficult obstacle encountered. After researching online and consulting classmates we were able to quickly find a way to initialize a keypad in the Arduino and how to have the Arduino read input and send output from the buttons pressed on the keypad. The biggest obstacle then was figuring out an algorithm to check if the passcode entered into the keypad was correct. Prior knowledge in other coding languages made this a rather simple task to accomplish, although more time consuming than we had hoped for, mainly due to translation issues, but sure enough it was something that was taken care of to work correctly with our keypad, sensors, and circuit. At the end of it, all obstacles faced were either worked around in a manner that did not impact the performance of the backpack lock or was dealt with and overcome completely.

Code

```
#include <Keyboard.h>
#include <Arduino.h>
#include <Keypad.h>

uint8_t outBit;
volatile uint8_t* out;
const byte ROWS = 4; // Four rows
const byte COLS = 3; // Three columns
String user = "";
// Define the Keymap
```

```
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
// Connect keypad ROW0, ROW1, ROW2 and ROW3 to these Arduino pins.
byte rowPins[ROWS] = {24,26,28,30}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {32,34,36}; //connect to the column pinouts of the keypad

// Create the Keypad
Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

#define ledpin 13

void setup()
{
  pinMode(ledpin,OUTPUT);
  digitalWrite(ledpin, HIGH);
  outBit = digitalPinToBitMask(ledpin);
  out = portOutputRegister(digitalPinToPort(ledpin));
  Serial.begin(9600);
}

void loop()
{
  char key = kpd.getKey();
  int ctr = ctr;
  int alarm = alarm;
  if(key)  // Check for a valid key.
  {
    switch (key)
    {
      case '*':
        digitalWrite(ledpin, LOW);
        break;
      case '#':
        digitalWrite(ledpin, HIGH);
        break;
```

```
    default:
    {
     Serial.println(key);
     user = user + "" + (String)key;
    }
  }// end switch
 }//end if
 if(key)
 {
  if(key=='*')
  {
  int alarm = 0;
  String pswd ="1234";
  int ctr = 0;
  while(ctr < 4)
      {
         String letter = user.substring(ctr, ctr+1);
         String act = pswd.substring(ctr, ctr+1);
         if(letter.equals(act))
             alarm = alarm;
         else
            alarm += 1;
         ctr++;
      }
      ctr = 0;
      user = "";
  /*if(key=='#')
    alarm = 0;*/
 if(alarm > 0)
 {*out &= ~outBit;//(ledpin, HIGH);
  delay(500);
 Serial.println("1");}
 else
 {*out |= outBit;//(ledpin, LOW);
  delay(500);
 Serial.println("0");}
  }
 }// end checker
}// end void loop
```