Skunks Misery 1.0 conference badge

Compass

Designed by Michael Vieau and Kevin Bong

Created on April 18, 2019

Last revised May 31, 2019

Table of Contents

Design Goal

The goal of the design was to make useful tool that would double as the conference badge. Since the Skunks Misery conference was an outdoor hacker conference, a tool that could be used in a woodland setting was selected.

Design Process

As with most of our designs, we started with a solderless breadboard and off the shelf components. After we work out the bugs and decide on which components we will be using, we start looking for the components in the correct package. For this board we are using the 1206 package.

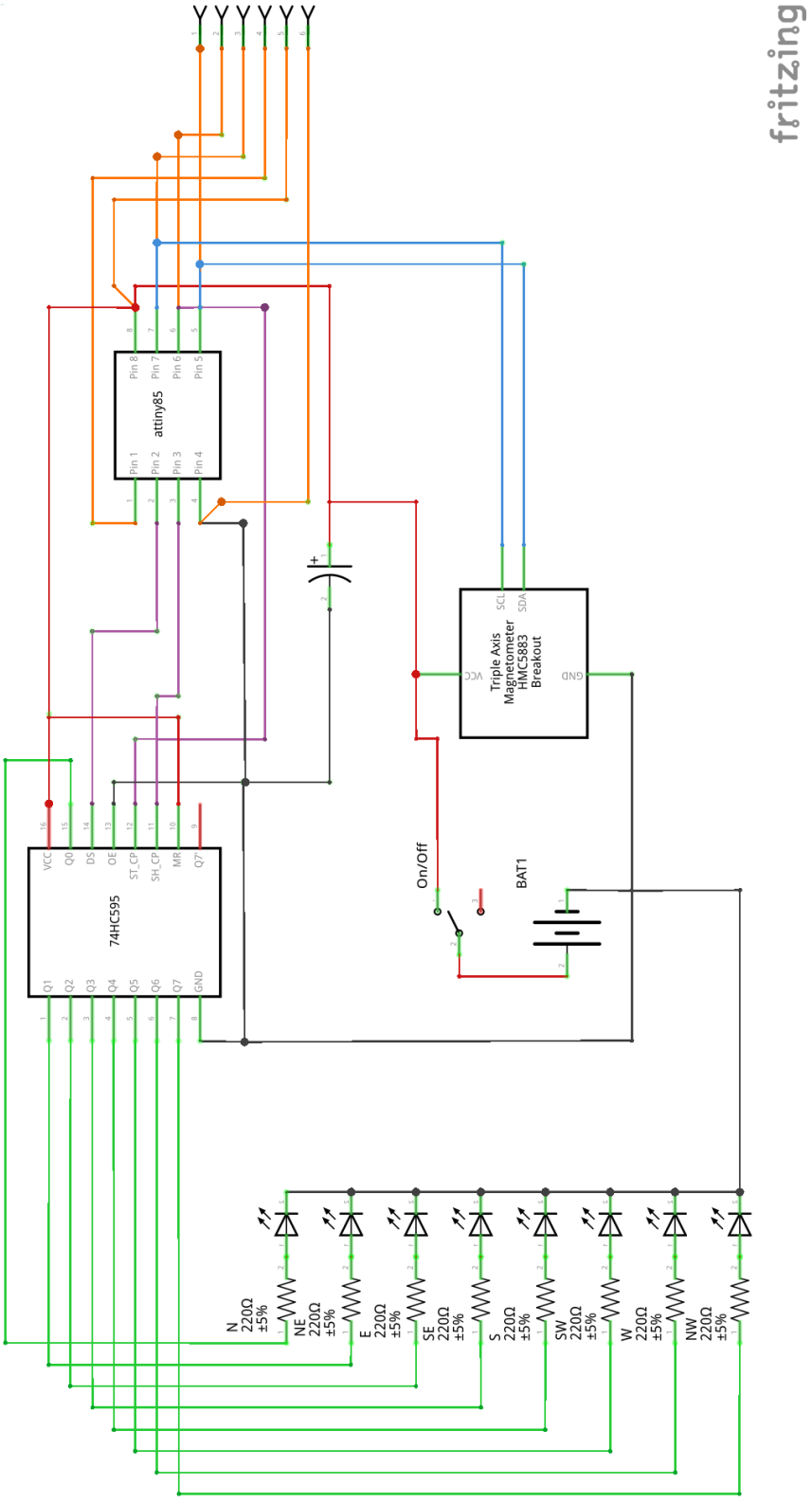
Parts List

The following table lists the parts required to build a version 1.0 Skunks Misery badge from the 2019 conference. The board is setup to use surface mount parts with the 1206 package.

Qty	Part Number	Description	Data Sheet
1	N/A	Main PCB	N/A
1	74HC595D	Shift register	https://assets.nexperia.com/documents/data-sheet/74HC_HCT595.pdf
1	ATTINY85-20SUR	Attiny85	http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85_Datasheet.pdf
1	HMC5883L	Magnetometer	https://media.digikey.com /pdf/Data%20Sheets/Honeywell%20PDFs/HMC5883L.pdf
1	GRM31CR61H475KA12L	4.7 µF Capacitor	https://www.mouser.com/datasheet/2/281/murata_03052018_GRM_Series_1-1310166.pdf
8	150120BS75000	LEDs	https://www.mouser.com/datasheet/2/445/150120BS75000-368826.pdf
8	ERJ-UP8F2200V	220 Ohm Resistor	https://www.mouser.com/datasheet/2/315/AOA0000C337-1276285.pdf
1	BAT-HLD-001	Battery holder	https://www.mouser.com/datasheet/2/238/bat-hld-001-220194.pdf
1	MSS22D18D	Switch (on-off)	
1	CR2032	Battery	http://data.energizer.com/pdfs/cr2032.pdf

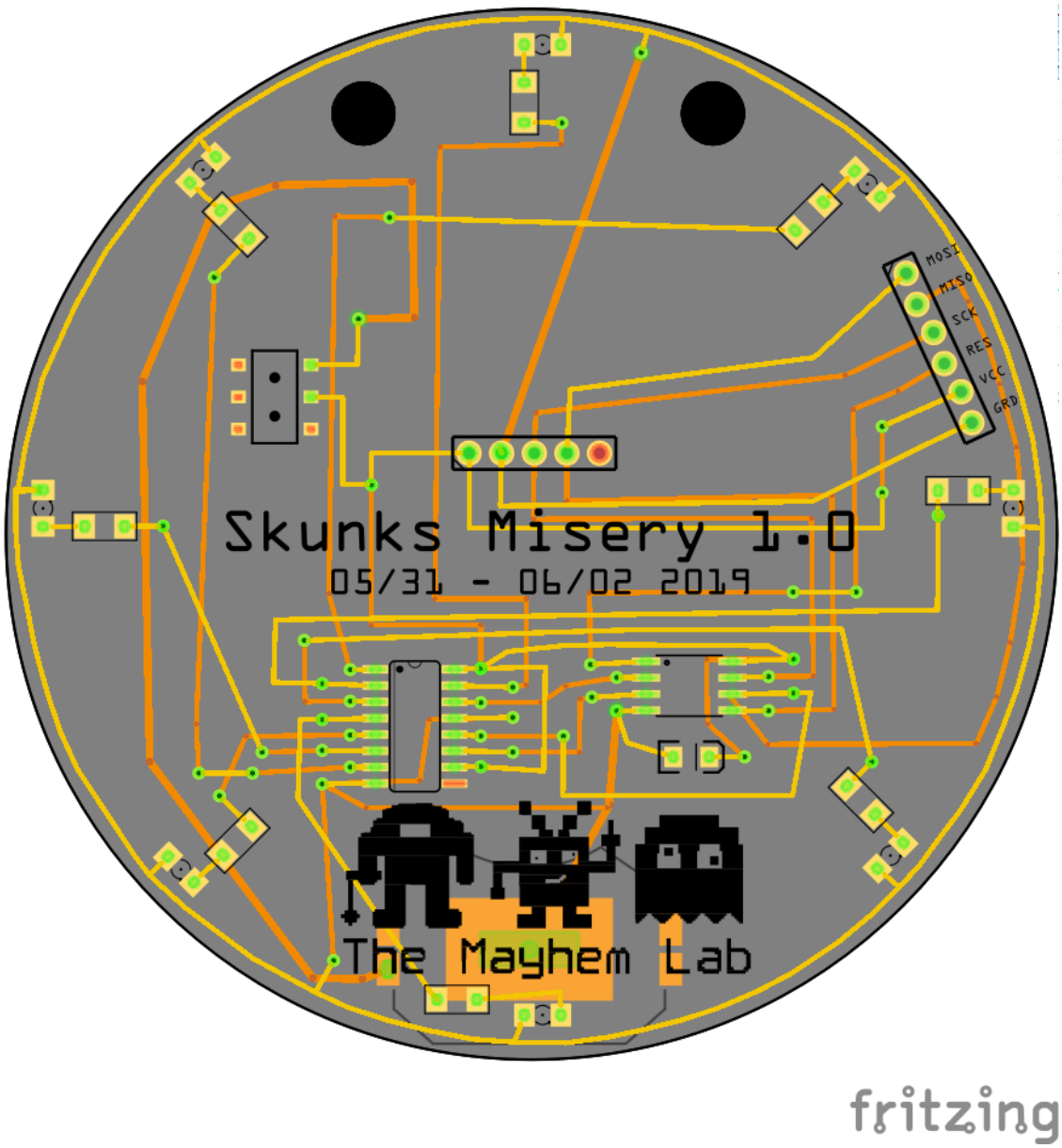
Schematics

The following is the logical electrical schematic layout of the Skunks Misery 1.0 conference badge.



PCB Layout

The PCBs were printed by PCBWay (https://www.pcbway.com). The following are images of the front and back of the PCB.

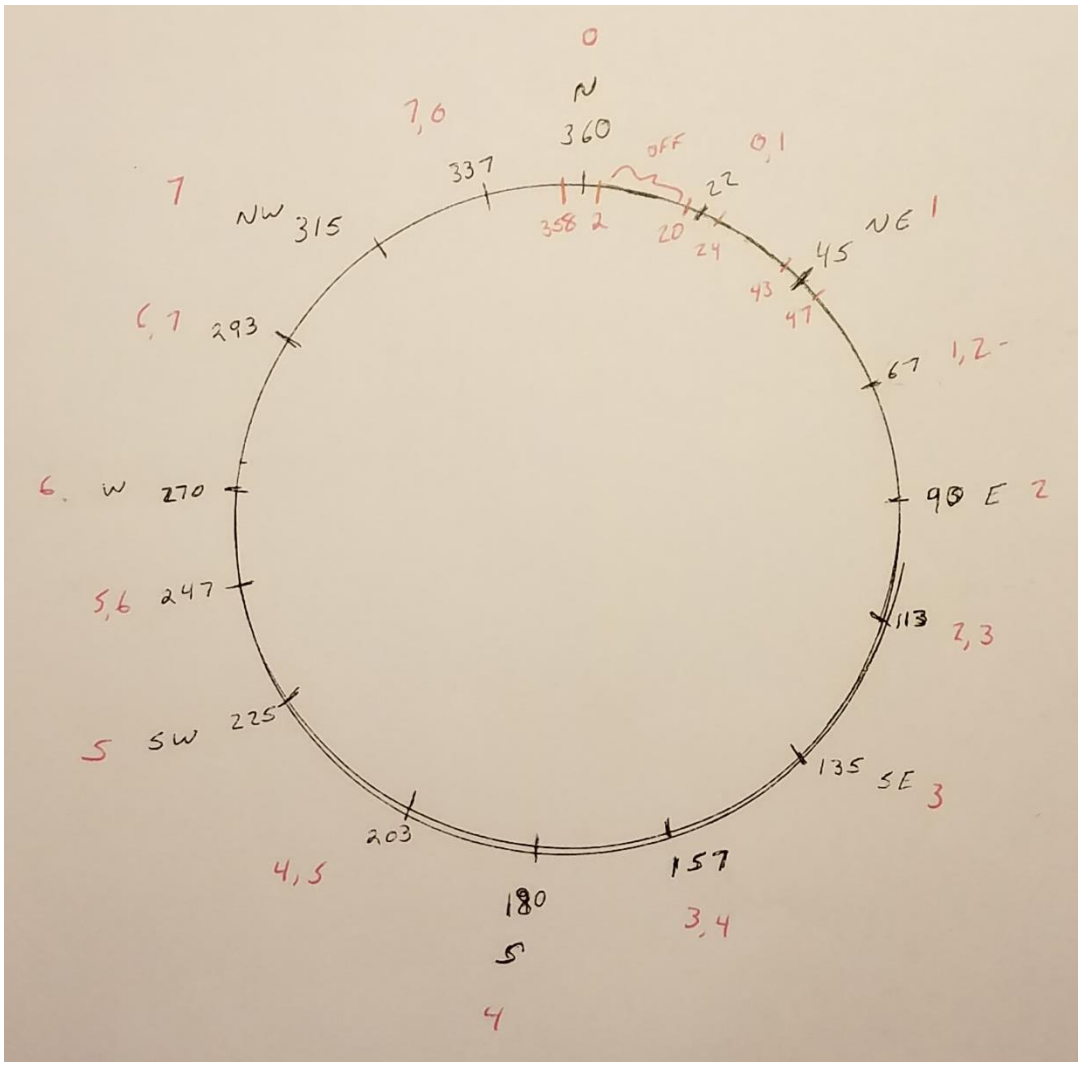


LED Math

The Magnetometer was very sensitive to different magnetic fields, so we spent time tuning when each LED would turn on. The board has eight LEDs and we needed to determine how much leeway we would build in when each LED would light up. The more leeway added would reduce the accuracy of the compass. On the other hand, if we only light the LED when it was pointed directly at the correct degree, the compass would be very difficult to use.

We went with a two-degree offset in either direction, resulting in a four degree 'window' when the LED would light up. As an example, the North LED would light up with the compass registering between 358 and 2 degrees.

It was decided to have two LEDs light up when the compass was pointed to the degree halfway between the corresponding LEDs. As an example, the North and Northeast LEDs would light up when the compass registered a degree between 20 and 24.



Code

The code was written using the Arduino IDE, version 1.8.6. The code requires two libraries for it to function. First, you will need the *wire* library in order to utilize the I2C protocol and communicate with the magnetometer. You will also need the MechaQMC5883 library in order to read data from the magnetometer. You will find a link to the MechaQMC5883 library under the

```
/*
 * Name:      Compass
 * Purpose:   Skunks Misery Badge 1.0 - Compass
 * By:        Micael Vieau & Kevin Bong
 * Created:   2019.4.16
 * Modified:  2019.5.20
 * Rev Level  1.0
 */

#include <Wire.h> //Wire Librarey for I2C communication
#include <HMC5883L.h>
#include <MechaQMC5883.h> //QMC5883 Library is added since mine is QMC583 and not HMC5883

MechaQMC5883 qmc; //Create an object name for the sensor, I have named it as qmc

char led_count = 7; // Total number of LEDs - 1
int startcount = 0; // Counter used for the startup sequence
int degreevariant = 2; // This is the number of degrees on each side of the direction we want to allow for the LED to be on.
//int degreevariant = 22.5;
// ****************** Shift register stuff ******************
int latchPin = 1; // Pin 6 (PB1) on the ATtiny85
int clockPin = 4; // Pin 3 (PB4) on the ATtiny85
int dataPin = 3;  // Pin 2 (PB3) on the ATtiny85
byte leds = B00000000; // LEDS off by default
int heading=0;

void setup() {
  Wire.begin(); //Begin I2C communication
  qmc.init(); //Initialise the QMC5883 Sensor

  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}

void loop() {

  // Run startup funciton
  if (startcount < 2)
```

```
{
  startup();
  AllOff();
}

int x,y,z;
qmc.read(&x,&y,&z); //Get the values of X,Y and Z from sensor
 int heading=atan2(x, y)/0.0174532925; //Calculate the degree using X and Y parameters with this formulae

//Convert result into 0 to 360
if(heading < 0)
{
  heading+=360;
}
heading = 360-heading;

//Based on the value of heading light up the correct LED(s).

// North
if (heading > (360-degreevariant) || heading < (0+degreevariant))
{
  OneOn(0);
}
// North by North-East
if (heading > (22-degreevariant) && heading < (22+degreevariant))
{
  TwoOn(7,0);
}
// North-East
if (heading > (45-degreevariant) && heading < (45+degreevariant))
{
  OneOn(7);
}
// North-East by East
if (heading > (67-degreevariant) && heading < (67+degreevariant))
{
  TwoOn(7,6);
}
// East
if (heading > (90-degreevariant) && heading < (90+degreevariant))
{
  OneOn(6);
}
// East by South-East
if (heading > (113-degreevariant) && heading < (113+degreevariant))
{
  TwoOn(6,5);
```

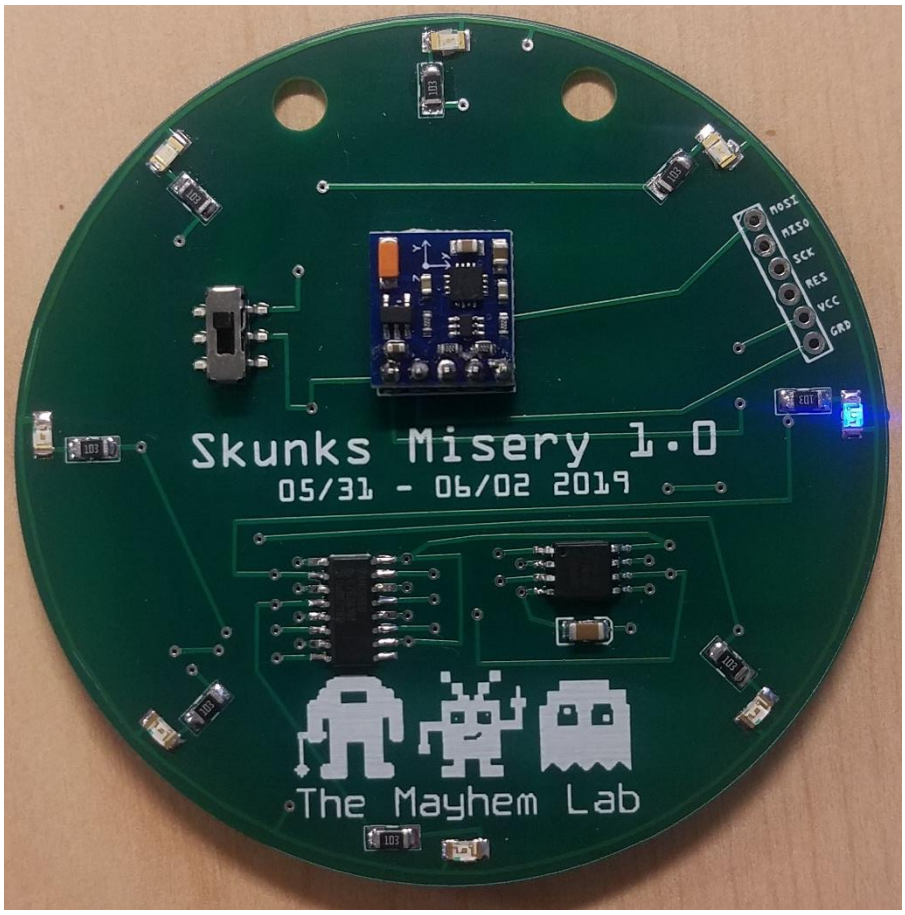
```
}
// South-East
if (heading > (135-degreevariant) && heading < (135+degreevariant))
{
  OneOn(5);
}
// South-East by South
if (heading > (157-degreevariant) && heading < (157+degreevariant))
{
  TwoOn(5,4);
}
// South
if (heading > (180-degreevariant) && heading < (180+degreevariant))
{
  OneOn(4);
}
// South by South-West
if (heading > (203-degreevariant) && heading < (203+degreevariant))
{
 TwoOn(4,3);
}
// South-West
if (heading > (225-degreevariant) && heading < (225+degreevariant))
{
  OneOn(3);
}
// South-West by West
if (heading > (247-degreevariant) && heading < (247+degreevariant))
{
  TwoOn(3,2);
}
// West
if (heading > (270-degreevariant) && heading < (270+degreevariant))
{
  OneOn(2);
}
// West by North-West
if (heading > (293-degreevariant) && heading < (293+degreevariant))
{
 TwoOn(2,1);
}
// North-West
if (heading > (315-degreevariant) && heading < (315+degreevariant))
{
  OneOn(1);
}
// North-West by North
if (heading > (337-degreevariant) && heading < (337+degreevariant))
```

```
 {
   TwoOn(1,0);
 }
 else
 {
   AllOff();
 }
}

void AllOff()
{
 // Turns off all the LEDs
 delay(5);

 for (int i = 0; i < 8; i++)
 {
   bitClear(leds, i);
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, MSBFIRST, leds);
   digitalWrite(latchPin, LOW);
 }
 delay(5);
}

void OneOn(int whichLED)
{
 // Turn on one LED when pointed in the correct direction
 leds = B00000000;
 delay(45);
   bitSet(leds, whichLED);
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, MSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
 //delay(5);
}

void TwoOn(int whichLED1, int whichLED2)
{
 // Turn on two LEDs when between two directions
 leds = B00000000;
 delay(5);
   bitSet(leds, whichLED1);
   bitSet(leds, whichLED2);
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, MSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
 delay(5);
}
```

```
void startup()
{
 // Function to cycle through the LEDs when the board turns on
 leds = B00000000;
 for (int i = 0; i < 8; i++)
 {
  OneOn(i);
  delay(50);
 }
 startcount++;
}
```

Completed Badge

After the completed badge is powered on, the LED(s) that are light will indicate a Northerly direction.



References

Mecha_QMC5883L Arduino Library
https://github.com/keepworking/Mecha_QMC5883L/blob/master/MechaQMC5883.h

Files for this build
https://github.com/TheMayhamLab/SkunksMisery-1.0-Badge