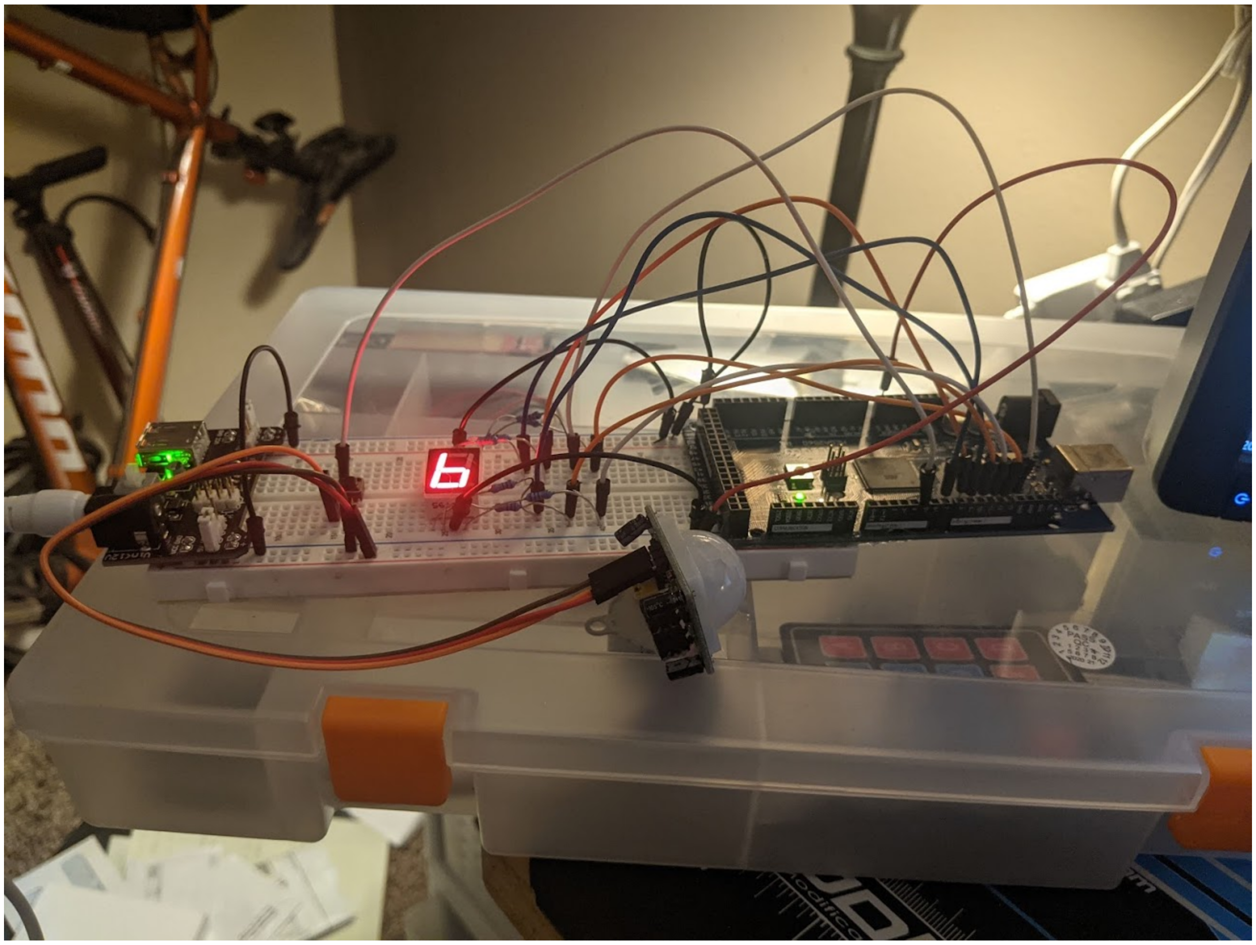
Lab 2 Report

Motion Sensor & 7-Segment Display

CENG 347 Embedded Intelligent Systems



Samuel Donovan & Samantha Pfeiffer

2021-02-04

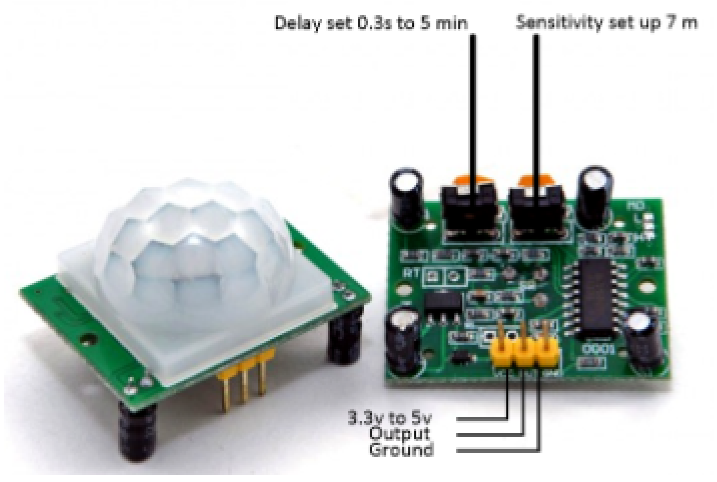
OVERVIEW

The purpose of this lab was to become more familiar with GPIO ports and learn how to interface with a motion sensor and the seven segment display.

--- MOTION SENSOR --

When the PIR motion sensor is triggered its output pin drives HIGH and stays HIGH for a number of seconds determined by the delay potentiometer. Delay times can range from 0.3 seconds to 5 minutes. The second potentiometer controls the distance at which motion can be detected; the furthest range being 7 meters.

Figure 1 PIR Motion Sensor



-- 7-SEGMENT DISPLAY --

7-segment displays are actually *eight* LEDs planted onto a surface in a pattern that allows users to construct various characters formed from LED patterns. While some displays are *common cathode*, the one we used was *common anode*, meaning that each LED pin must be driven HIGH to luminate it, while the common pin should be tied to ground.

While the displayable characters are numerous, for our purposes we required only decimal digits 0-9. **Table 1** shows which LEDs should be luminated when.

Figure 2 7-Segment Display LED Names

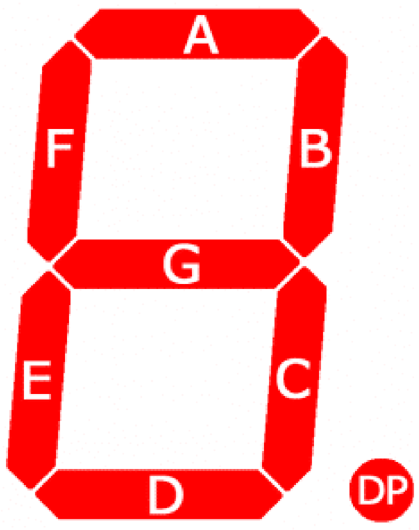


Table 1 7-Segment Truth Table

Decimal Value	A	B	C	D	E	F	G
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	0	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	0	0	1	1

METHODS

Each member worked with circuitry and code. The motion sensor was configured by Sam Pfeiffer and the 7-segment display by Sam Donovan. Combining the two was completed collectively.

--- CIRCUITRY ---

This circuit was built using an ATMega 2560k controller board, a bread board, a 5v power bus, a PIR motion sensor, a 7-segment display, 7 330 Ohm resistors, and several connecting wires.

The motion sensor was driven by the 5v power source and its output was connected to **PE3 (Pin 6)**.

The 7-segment display was connected according to the pattern in **Table 2**. The LED names and Pin numbers are available in **Figure 2** and **Appendix A**. Each segment was driven by the ATMega and current limited by a 330 Ohm resistor. The common anode pin was tied to ground.

Table 2 7-Segment Wiring Table

LED	Port	Pin
A	PB7	13
B	PB6	12
C	PB5	11
D	PB4	10
E	PH6	9
F	PH5	8
G	PH4	7
DP	/	/

--- CODE ---

Expanding on our code used in **Lab 1**, two data types were made to ease passing the information around between functions: `struct GPIO_Pin` and `struct SegDisplay`. As is evident, `GPIO_Pin` represents a single GPIO Pin on the board, and holds it's DDR (`ddr`), PORT-OUT (`write`), and PORT-IN (`read`) addresses, as well as the desired pin (`pin`). `SegDisplay` contains eight `GPIO_Pin` members, each representing a pin connected to a 7-segment display. The member names correlate with the LEDs shown in **Figure 2**.

For GPIO manipulation, three functions, `setPinMode`, `setPinState` and `getPinState` exist. For the 7-segment, we wrote one function `setupSegDisplay`, to set up the pins to be outputs in a LOW state. The other function, `showSegDisplay`, acts as a driver to abstract the process of writing a single-digit decimal number to the display. While this function is lengthy and could have been written in a more clever way, we believe in its current state it's legibility and maintainability exonerate it.

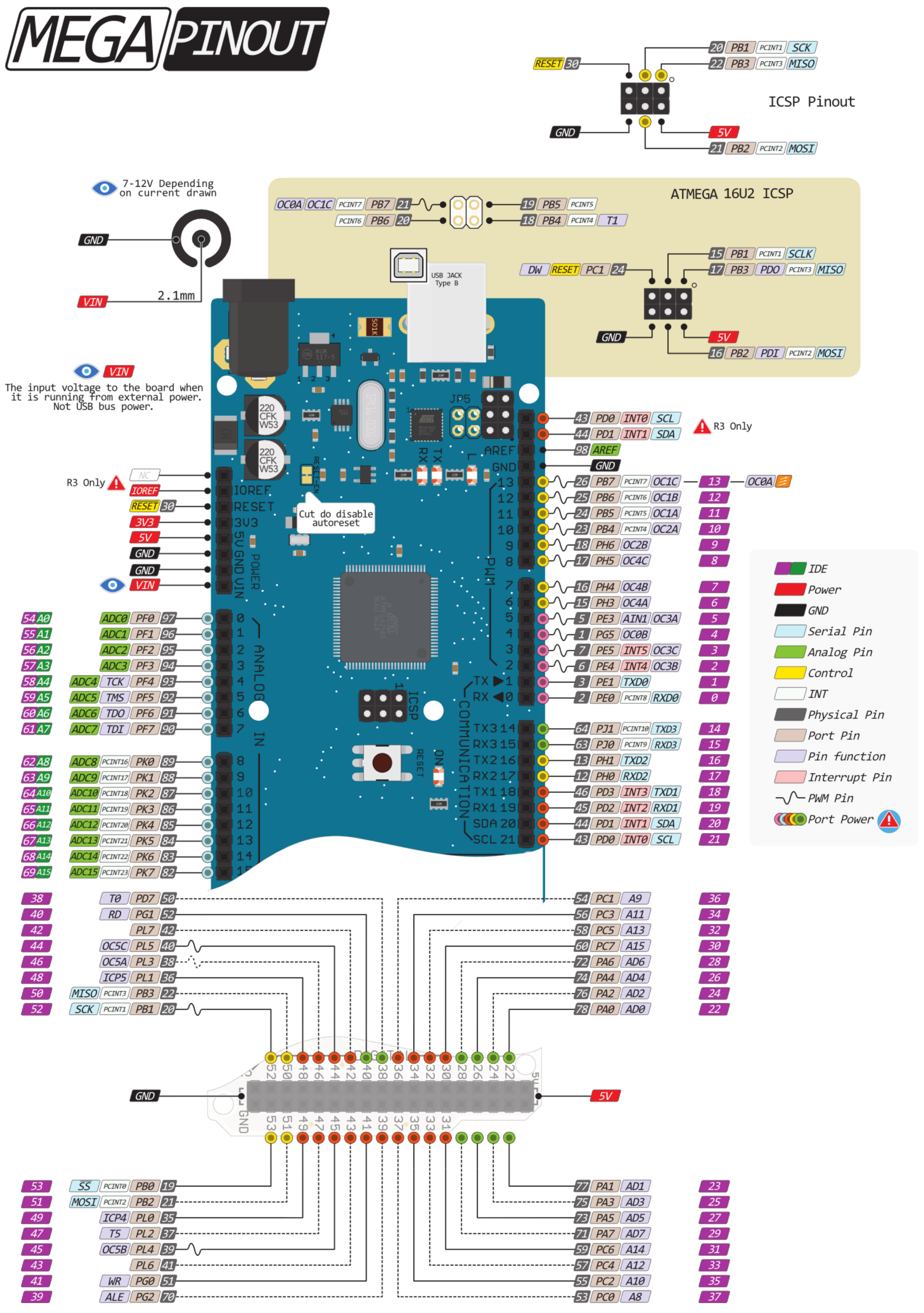
All code is present in **Appendix B**.

RESULTS

The circuit worked as expected for the most part. The display increments when motion is detected. Once the count exceeds 9, it resets to 0.

One issue was noticed in the behavior of the motion sensor. Once triggered, there is a 5 second delay before it can trigger again. This seems to remain no matter what the delay potentiometer is set to. Additionally, the motion sensor produces false positives and true negatives when positioned at a very close distance to where it's meant to detect.

APPENDIX A: ATMega 2560 Pinout



APPENDIX B: Motion Sensor and 7-Segment Display C Code

```
/*   Filename : Lab2.ino
 *   Author(s): Samuel Donovan, Samantha Pfeiffer
 *   Config: ATmega2560 @16MHz
 *   Description: Increments a 7-Segment display on motion
sensor trigger
 */
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
typedef volatile unsigned char gpioreg;

struct GPIO_Pin
{
  gpioreg* ddr;
  gpioreg* write;
  gpioreg* read;
  byte pin;
};
struct SegDisplay
{
  GPIO_Pin A;
  GPIO_Pin B;
  GPIO_Pin C;
  GPIO_Pin D;
  GPIO_Pin E;
  GPIO_Pin F;
  GPIO_Pin G;
  GPIO_Pin DP;
};

//Pin Functions
void setPinMode(GPIO_Pin PIN, byte MODE)
{
  if(MODE == INPUT)
    *PIN.ddr &= ~(1 << PIN.pin);
  else
    *PIN.ddr |= 1 << PIN.pin;
}
void setPinState(GPIO_Pin PIN, byte STATE)
```

```
{
  if(STATE == LOW)
    *PIN.write &= ~(1 << PIN.pin);
  else
    *PIN.write |= 1 << PIN.pin;
}
byte getPinState(GPIO_Pin PIN)
{
  if(*PIN.read >> PIN.pin)
    return HIGH;
  else
    return LOW;
}

//Seg Display Functions
void setupSegDisplay(SegDisplay disp)
{
  setPinMode(disp.A, OUTPUT);
  setPinMode(disp.B, OUTPUT);
  setPinMode(disp.C, OUTPUT);
  setPinMode(disp.D, OUTPUT);
  setPinMode(disp.E, OUTPUT);
  setPinMode(disp.F, OUTPUT);
  setPinMode(disp.G, OUTPUT);
  setPinMode(disp.DP, OUTPUT);
}
void showSegDisplay(byte b, SegDisplay disp, bool dp=false)
{
  if(dp)
    setPinState(disp.DP, HIGH);
  else
    setPinState(disp.DP, LOW);

  switch(b)
  {
    case 0x0:
      setPinState(disp.A, HIGH);
      setPinState(disp.B, HIGH);
      setPinState(disp.C, HIGH);
      setPinState(disp.D, HIGH);
      setPinState(disp.E, HIGH);
      setPinState(disp.F, HIGH);
      setPinState(disp.G, LOW);
      break;
    case 0x1:
      setPinState(disp.A, LOW);
      setPinState(disp.B, HIGH);
      setPinState(disp.C, HIGH);
      setPinState(disp.D, LOW);
      setPinState(disp.E, LOW);
      setPinState(disp.F, LOW);
      setPinState(disp.G, LOW);
      break;
    case 0x2:
      setPinState(disp.A, HIGH);
      setPinState(disp.B, HIGH);
      setPinState(disp.C, LOW);
      setPinState(disp.D, HIGH);
      setPinState(disp.E, HIGH);
      setPinState(disp.F, LOW);
      setPinState(disp.G, HIGH);
      break;
    case 0x3:
      setPinState(disp.A, HIGH);
      setPinState(disp.B, HIGH);
      setPinState(disp.C, HIGH);
      setPinState(disp.D, HIGH);
      setPinState(disp.E, LOW);
      setPinState(disp.F, LOW);
      setPinState(disp.G, HIGH);
      break;
    case 0x4:
      setPinState(disp.A, LOW);
      setPinState(disp.B, HIGH);
      setPinState(disp.C, HIGH);
      setPinState(disp.D, LOW);
      setPinState(disp.E, LOW);
      setPinState(disp.F, HIGH);
      setPinState(disp.G, HIGH);
      break;
    case 0x5:
      setPinState(disp.A, HIGH);
      setPinState(disp.B, LOW);
      setPinState(disp.C, HIGH);
      setPinState(disp.D, HIGH);
      setPinState(disp.E, LOW);
      setPinState(disp.F, HIGH);
      setPinState(disp.G, HIGH);
      break;
    case 0x6:
      setPinState(disp.A, LOW);
      setPinState(disp.B, LOW);
      setPinState(disp.C, HIGH);
      setPinState(disp.D, HIGH);
      setPinState(disp.E, HIGH);
      setPinState(disp.F, HIGH);
      setPinState(disp.G, HIGH);
      break;
    case 0x7:
      setPinState(disp.A, HIGH);
      setPinState(disp.B, HIGH);
      setPinState(disp.C, HIGH);
      setPinState(disp.D, LOW);
      setPinState(disp.E, LOW);
      setPinState(disp.F, LOW);
      setPinState(disp.G, LOW);
      break;
    case 0x8:
      setPinState(disp.A, HIGH);
      setPinState(disp.B, HIGH);
      setPinState(disp.C, HIGH);
      setPinState(disp.D, HIGH);
      setPinState(disp.E, HIGH);
      setPinState(disp.F, HIGH);
      setPinState(disp.G, HIGH);
      break;
    case 0x9:
      setPinState(disp.A, HIGH);
      setPinState(disp.B, HIGH);
      setPinState(disp.C, HIGH);
      setPinState(disp.D, LOW);
      setPinState(disp.E, LOW);
      setPinState(disp.F, HIGH);
      setPinState(disp.G, HIGH);
      break;
    default:
      setPinState(disp.A, LOW);
      setPinState(disp.B, LOW);
      setPinState(disp.C, LOW);
      setPinState(disp.D, LOW);
      setPinState(disp.E, LOW);
      setPinState(disp.F, LOW);
      setPinState(disp.G, LOW);
      break;
  }
}

int main()
{
  // Vars
  SegDisplay disp =
  {
    A: {&DDRB, &PORTB, NULL, 7},
    B: {&DDRB, &PORTB, NULL, 6},
    C: {&DDRB, &PORTB, NULL, 5},
    D: {&DDRB, &PORTB, NULL, 4},
    E: {&DDRH, &PORTH, NULL, 6},
    F: {&DDRH, &PORTH, NULL, 5},
    G: {&DDRH, &PORTH, NULL, 4},
    DP:{NULL, NULL, NULL, NULL}
  };
  GPIO_Pin snsr = {&DDRE, NULL, &PINE, 3};
  byte n = 0;
  byte prevState = LOW;

  //Setup
  setupSegDisplay(disp);
  setPinMode(snsr, INPUT);
  showSegDisplay(n, disp);

  //Loop
  while(1)
  {
    if(getPinState(snsr) == HIGH)
    {
      if(prevState == LOW)
      {
        n++;
        if(n > 9)
          n = 0;

        showSegDisplay(n, disp);
      }
      prevState = HIGH;
    }
    else
    {
      prevState = LOW;
    }
  }
}
```