CENG 347L Arduino IDE and GPIO

Laboratory 1

Getting Started, GPIO, and Debugging

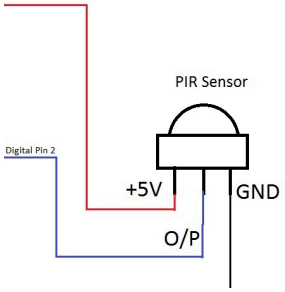
Zackery Holloway/ Alexis Englund 2/06/25

Purpose:

1. Integrating a Passive Infrared (PIR) motion sensor to monitor activity
2. Set up a 7-segment display to count from 0-9.
3. Integrate the PIR with 7 segments to count when triggered.

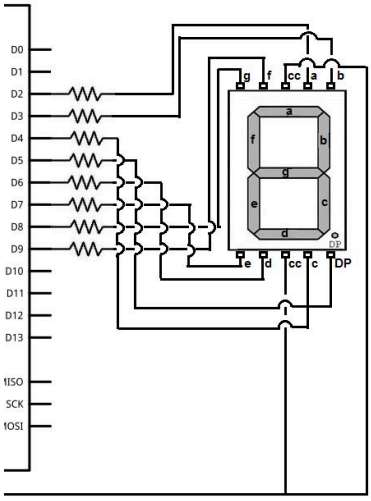
Procedure:

The lab is divided into three key parts:



Part 1: Setting Up the PIR Sensor

1. Connect the PIR sensor to ground, 5V, and PB6
2. Use the onboard LED and set PB7 as the output
3. Adjust the potentiometers to get a valid value.



Part 2: 7 Segemnet Counter Implementation

1. Connect the 7 Segment display using 6 pins and the two common grounds.
2. Set up the ports in the code.
3. Write a while loop to count from 0 to 9, to test functionality.

Part 3: Integration

1. Create a variable count to increment the counter.
2. Change the logic such that when PB6 is high it increments the counter by 1.
3. Add logic so that when the counter is larger than 10, it resets to 0.

CENG 347L Arduino IDE and GPIO

Laboratory 1

Getting Started, GPIO, and Debugging

Zackery Holloway/ Alexis Englund 2/06/25

Application:

Part 1: Setting Up the PIR Sensor

The code for this was fairly simple just setting up the inputs and outputs. Working with the sensor had issues but nothing that could not be fixed. At times it was easiest just to hook a high pin up to check it the system was working properly.

Code:

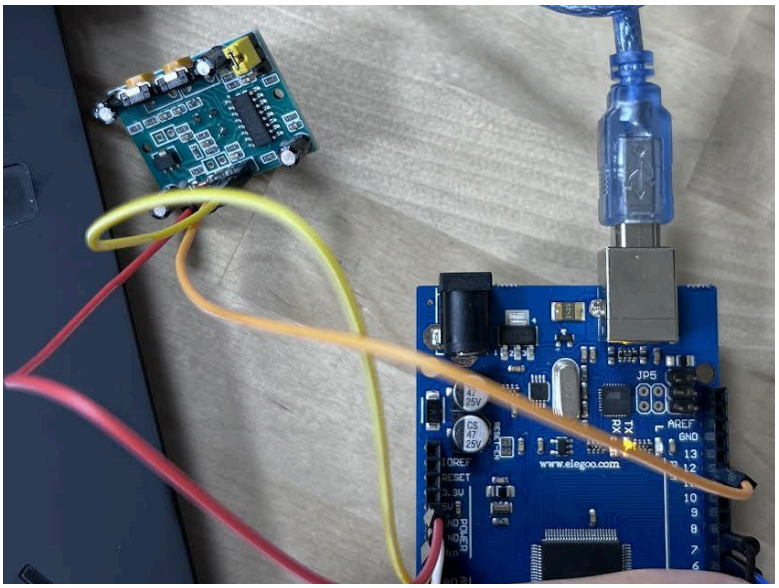
```
#include <avr/io.h>
#include <util/delay.h>

int main()
{

  DDRB &= ~(1 << PB2);// Clear PD2 (pin 2) bit in DDRD to set
it as input

  DDRB |= (1 << PB7);//setup PORTB (pin7) as an output

  //set PORTB to a known state
  //PORTB = 0xFF;

  //loop for continuous blinking
  while(true)
  {
    if(PINB & (1 << PB2))// Check if pin 2 is HIGH
    {
      PORTB |= (1 << PB7);// Sets LED as HIGH
    }
    else
    {
      PORTB &= ~(1 << PB7);// Sets LED as LOW
    }
  }
}
```

CENG 347L Arduino IDE and GPIO

Laboratory 1

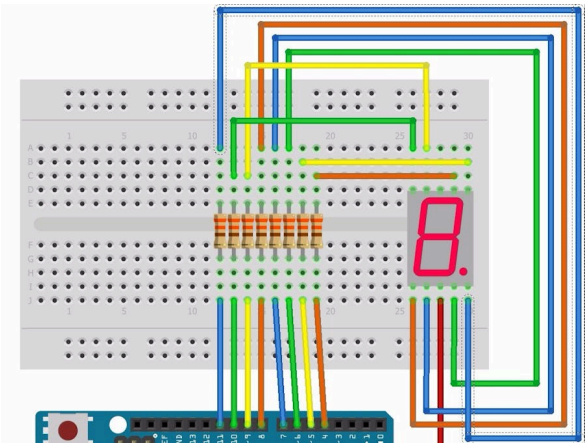
Getting Started, GPIO, and Debugging

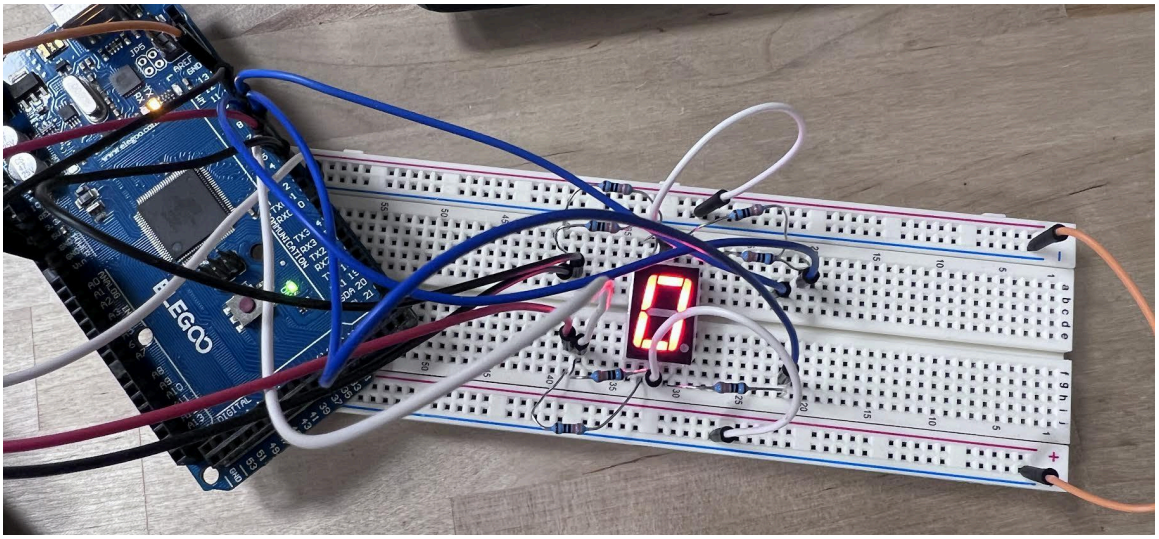
Zackery Holloway/ Alexis Englund 2/06/25



Part 2: 7 Segemnet Counter Implementation

The diagram below shows how the circuit was wired.





CENG 347L Arduino IDE and GPIO

Laboratory 1

Getting Started, GPIO, and Debugging

Zackery Holloway/ Alexis Englund 2/06/25

The following code was written giving the 7 segment its function to turn on and off each segment instead of having it in main. A count was added to increment the counter.

Code:

```
#include <avr/io.h>
#include <util/delay.h>

int count = 0;

int main()
{
  Serial.begin(9600);
  DDRB &= ~(1 << PB6); // Clear PD2 (pin 2) bit in DDRD to set it as input

  DDRB |= (1 << PB7);          //setup PORTB (pin7) as an output

  //7Segment
  DDRH |= (1 << PH4); //A
  DDRH |= (1 << PH3); //B
  DDRE |= (1 << PE3); //C
  DDRG |= (1 << PG5); //D
  DDRE |= (1 << PE5); //E
  DDRE |= (1 << PE4); //F
  DDRH |= (1 << PH5); //G

  //set PORTB to a known state
  //PORTB = 0xFF;

  seg(count);           //Updates 7 segment

  bool prevState = false; // Variable to store previous state

  while(true)                  //loop for continuous update checking
  {
      Count++;
```

Zackery Holloway/ Alexis Englund 2/06/25

```
      if(count < 10)
      {
          seg(count);
      }
      else
      {
          count = 0;
          seg(count);
  }

void seg(int count)
{
  //Clear Ports
  PORTH &= ~(1 << PH4); //A
  PORTH &= ~(1 << PH3); //B
  PORTE &= ~(1 << PE3); //C
  PORTG &= ~(1 << PG5); //D
  PORTE &= ~(1 << PE5); //E
  PORTE &= ~(1 << PE4); //F
  PORTH &= ~(1 << PH5); //G
  Serial.println("Make it here");
  if(count == 0)
  {
    PORTH |= (1 << PH4); //A
    PORTH |= (1 << PH3); //B
    PORTE |= (1 << PE3); //C
    PORTG |= (1 << PG5); //D
    PORTE |= (1 << PE5); //E
    PORTE |= (1 << PE4); //F
    Serial.println("Make it here 1");
  }
  else if (count == 1)
  {
    PORTH |= (1 << PH3); //B
    PORTE |= (1 << PE3); //C
  }
  else if (count == 2)
```

Zackery Holloway/ Alexis Englund 2/06/25

```
{
  PORTH |= (1 << PH4); //A
  PORTH |= (1 << PH3); //B
  PORTG |= (1 << PG5); //D
  PORTE |= (1 << PE5); //E
  PORTH |= (1 << PH5); //G
}
else if (count == 3)
{
  PORTH |= (1 << PH4); //A
  PORTH |= (1 << PH3); //B
  PORTE |= (1 << PE3); //C
  PORTG |= (1 << PG5); //D
  PORTH |= (1 << PH5); //G
}
else if (count == 4)
{
  PORTH |= (1 << PH3); //B
  PORTE |= (1 << PE3); //C
  PORTE |= (1 << PE4); //F
  PORTH |= (1 << PH5); //G
}
else if (count == 5)
{
  PORTH |= (1 << PH4); //A
  PORTE |= (1 << PE3); //C
  PORTG |= (1 << PG5); //D
  PORTE |= (1 << PE4); //F
  PORTH |= (1 << PH5); //G
}
else if (count == 6)
{
  PORTH |= (1 << PH4); //A
  PORTE |= (1 << PE3); //C
  PORTG |= (1 << PG5); //D
  PORTE |= (1 << PE5); //E
  PORTE |= (1 << PE4); //F
```

Zackery Holloway/ Alexis Englund 2/06/25

```
    PORTH |= (1 << PH5); //G
  }
  else if (count == 7)
  {
    PORTH |= (1 << PH4); //A
    PORTH |= (1 << PH3); //B
    PORTE |= (1 << PE3); //C
  }
  else if (count == 8)
  {
    PORTH |= (1 << PH4); //A
    PORTH |= (1 << PH3); //B
    PORTE |= (1 << PE3); //C
    PORTG |= (1 << PG5); //D
    PORTE |= (1 << PE5); //E
    PORTE |= (1 << PE4); //F
    PORTH |= (1 << PH5); //G
  }
  else
  {

    PORTH |= (1 << PH4); //A
    PORTH |= (1 << PH3); //B
    PORTE |= (1 << PE3); //C
    PORTE |= (1 << PE4); //F
    PORTH |= (1 << PH5); //G
  }
}
```

CENG 347L Arduino IDE and GPIO

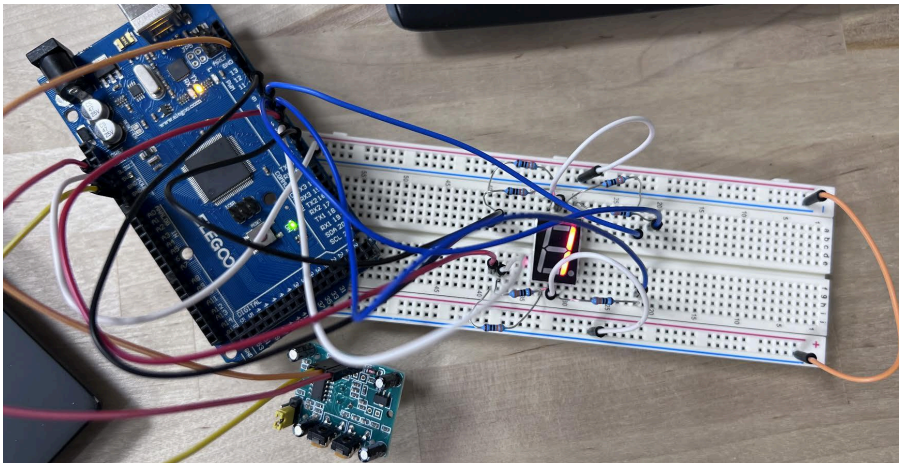
Laboratory 1

Getting Started, GPIO, and Debugging

Zackery Holloway/ Alexis Englund 2/06/25

Part 3: Integration

Finally, we integrated the code so that when it had an input signal it incremented the counter and updated the 7-segment.



The code was updated and attached below:

Code:

```
#include <avr/io.h>
#include <util/delay.h>

int count = 0;

int main()
{
  Serial.begin(9600);
  DDRB &= ~(1 << PB6);          // Clear PD2 (pin 2) bit in DDRD to set it
as input

  DDRB |= (1 << PB7);          //setup PORTB (pin7) as an output

  //7Segment
  DDRH |= (1 << PH4); //A
  DDRH |= (1 << PH3); //B
  DDRE |= (1 << PE3); //C
  DDRG |= (1 << PG5); //D
  DDRE |= (1 << PE5); //E
```

CENG 347L Arduino IDE and GPIO

Laboratory 1

Getting Started, GPIO, and Debugging

Zackery Holloway/ Alexis Englund 2/06/25

```
DDRE |= (1 << PE4); //F
DDRH |= (1 << PH5); //G

//set PORTB to a known state
//PORTB = 0xFF;

seg(count);           //Updates 7 segment

bool prevState = false; // Variable to store previous state

while(true)                  //loop for continuous update checking
{
  Serial.println(count);
  bool currentState = PINB & (1 << PB6);

  if(currentState && !prevState)    // Check if pin 6 is HIGH
  {
    PORTB |= (1 << PB7);     // Sets 7 HIGH
    count++;                 //Updates Count
    Serial.println(count);
    if(count < 10 )          //Checks boundary
    {
      Serial.println(count);
      seg(count);           //Updates 7 segment
    }
    else
    {
      Serial.println(count);
      count = 0;             //Resets count to 0
      seg(count);           //Updates 7 segment
    }
  }
  else
  {
    PORTB &= ~(1 << PB7);    //Sets 7 LOW
  }
  prevState = currentState;
```

Zackery Holloway/ Alexis Englund 2/06/25

```
  }
}

void seg(int count)
{
  //Clear Ports
  PORTH &= ~(1 << PH4); //A
  PORTH &= ~(1 << PH3); //B
  PORTE &= ~(1 << PE3); //C
  PORTG &= ~(1 << PG5); //D
  PORTE &= ~(1 << PE5); //E
  PORTE &= ~(1 << PE4); //F
  PORTH &= ~(1 << PH5); //G
  Serial.println("Make it here");
  if(count == 0)
  {
    PORTH |= (1 << PH4); //A
    PORTH |= (1 << PH3); //B
    PORTE |= (1 << PE3); //C
    PORTG |= (1 << PG5); //D
    PORTE |= (1 << PE5); //E
    PORTE |= (1 << PE4); //F
    Serial.println("Make it here 1");
  }
  else if (count == 1)
  {
    PORTH |= (1 << PH3); //B
    PORTE |= (1 << PE3); //C
  }
  else if (count == 2)
  {
    PORTH |= (1 << PH4); //A
    PORTH |= (1 << PH3); //B
    PORTG |= (1 << PG5); //D
    PORTE |= (1 << PE5); //E
    PORTH |= (1 << PH5); //G
  }
```

Zackery Holloway/ Alexis Englund 2/06/25

```
  else if (count == 3)
  {
    PORTH |= (1 << PH4); //A
    PORTH |= (1 << PH3); //B
    PORTE |= (1 << PE3); //C
    PORTG |= (1 << PG5); //D
    PORTH |= (1 << PH5); //G
  }
  else if (count == 4)
  {
    PORTH |= (1 << PH3); //B
    PORTE |= (1 << PE3); //C
    PORTE |= (1 << PE4); //F
    PORTH |= (1 << PH5); //G
  }
  else if (count == 5)
  {
    PORTH |= (1 << PH4); //A
    PORTE |= (1 << PE3); //C
    PORTG |= (1 << PG5); //D
    PORTE |= (1 << PE4); //F
    PORTH |= (1 << PH5); //G
  }
  else if (count == 6)
  {
    PORTH |= (1 << PH4); //A
    PORTE |= (1 << PE3); //C
    PORTG |= (1 << PG5); //D
    PORTE |= (1 << PE5); //E
    PORTE |= (1 << PE4); //F
    PORTH |= (1 << PH5); //G
  }
  else if (count == 7)
  {
    PORTH |= (1 << PH4); //A
    PORTH |= (1 << PH3); //B
    PORTE |= (1 << PE3); //C
```

CENG 347L Arduino IDE and GPIO

Laboratory 1

Getting Started, GPIO, and Debugging

Zackery Holloway/ Alexis Englund 2/06/25

```
  }
  else if (count == 8)
  {
    PORTH |= (1 << PH4); //A
    PORTH |= (1 << PH3); //B
    PORTE |= (1 << PE3); //C
    PORTG |= (1 << PG5); //D
    PORTE |= (1 << PE5); //E
    PORTE |= (1 << PE4); //F
    PORTH |= (1 << PH5); //G
  }
  else
  {

    PORTH |= (1 << PH4); //A
    PORTH |= (1 << PH3); //B
    PORTE |= (1 << PE3); //C
    PORTE |= (1 << PE4); //F
    PORTH |= (1 << PH5); //G
  }
}
```

Results:

Part 1: Setting Up the PIR Sensor

The LED turned on when the pin was high. Demonstrating that the PID was working correctly, although tuning was not the best.

Part 2: 7-Segment Counter Implementation

The loop correctly demonstrated the counting for 0-9. Turning on each segment. As it counted.

Part 3: Integration

The code was demonstrated correctly to the TA and when motion was detected or the pin was set to HIGH the counter incremented and then reset back to 0 at the end.

CENG 347L Arduino IDE and GPIO

Laboratory 1

Getting Started, GPIO, and Debugging

Zackery Holloway/ Alexis Englund 2/06/25

Conclusion:

In this lab, we successfully integrated a Passive Infrared (PIR) motion sensor with a 7-segment display to create a motion-triggered counter. The lab was divided into three key parts: configuring the PIR sensor, implementing the 7-segment display counter, and integrating both components to function as a cohesive system.

During the PIR sensor setup, we established the necessary input and output connections and verified functionality using an onboard LED. Although some minor issues arose with sensor sensitivity, they were resolved through manual testing and adjusting potentiometers.

For the 7-segment display, we implemented a program that counted from 0 to 9, ensuring proper segment control and number display. The integration step then combined the PIR sensor with the counter logic, allowing the display to increment whenever the sensor detected motion. A reset condition was also implemented to return the count to zero after reaching 9.

Overall, the system functioned as expected, effectively demonstrating sensor-driven counting. Future improvements could include refining the motion detection logic to reduce false triggers or implementing debounce techniques for more stable readings. This lab provided valuable hands-on experience with microcontroller interfacing, digital logic, and embedded system development.