Final Project – Motion Detected Lamp

Created By: Cody Morse

What I did:

My project is a LED lamp with the added capability of motion detection to turn on the lamp. When Professor Thompson brought up the ideas of adding capabilities changing current commercialized products in the beginning of the course, I immediately was thinking of items in my room I could alter in some way change its use. At first, I was thinking about altering a Keurig coffee maker, or doing something with my LED strips I have for my gaming setup. Those are items I didn't feel comfortable possibly screwing up, so I decided to alter my LED lamp. During class periods I would ponder of capabilities/features I could add to the LED lamp, and then I decided to add motion sensing feature to it. There isn't really much use to adding motion sensing to a lamp, but I thought it would be a cool idea.

The problem I tried to solve with my project varies because maybe you need light for just a couple of seconds, which is how I implemented mine, or if you need it for elongated periods of time. My finished product is an LED lamp with motion detection that keeps the lamp on for approximately three seconds.

There are always easier ways to do these kinds of projects like with an Arduino, which I initially used with my parts to make a strobe light. I put maybe twenty minutes of coding to make the project work, and there's no satisfaction taking the easy route when there's a harder path like using a microcontroller. I personally enjoyed working with microcontroller for this project because it requires you to use everything required to make a component work.

How I did it:

My project was designed using these parts: analog motion sensor (HC-SR501-PIR), two linear voltage regulators (5V 100mA and 3.3V 800mA), 5V relay (SRD-05VDC-SL-C), LED Lamp, 9V battery, and a MC9S08QG8 microcontroller. This is all shown in *figure 1* and *figure 3.* The overall perspective of my code was to send the ADC reading that is between a

certain range from the PIR motion sensor (3.3V) to the microcontroller (pin 16 – ADC pin) and make sure the reading is between 3.0V and 3.3V. Once the reading is compared to my designated range, if it falls within the range the microcontroller will send an output high signal to pin 15 (PTAD output), which will allow the relay to turn on. When the relay is turned on with the signal from the microcontroller the lamp will turn on, then turn off after three seconds because how my code is setup.

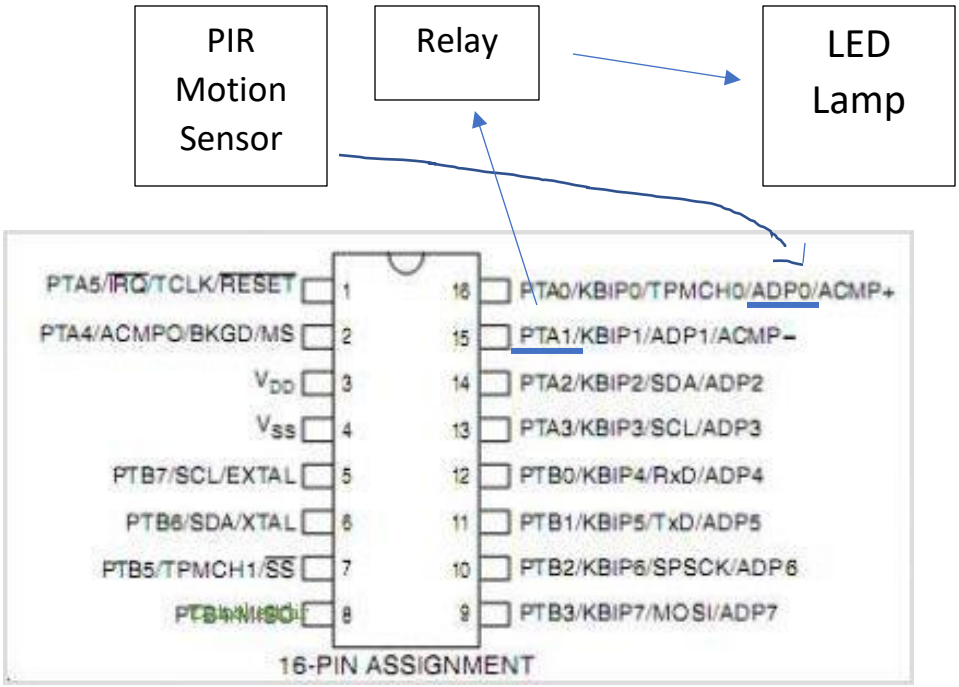


Figure 1.1 Diagram of hardware setup

The project itself was not too hard to implement, but I did create the project in assembly and realized C was much easier. Although assembly is our primary used coding language, I felt like C would be easier to implement for my project and I learned more using C rather than using assembly. My only real issue in the project was not realizing at the beginning that the PIR motion sensor was analog so my code wasn't working no matter what I did. From this mistake, I learned to inspect my parts before implementing any code since I don't like to make the same mistake twice.

How I know it worked:

When my project ended up working, I was so relieved. My goal for the project was for it to detect motion, turn on the light, and keep the light on for approximately three seconds. My code is designed to keep the light on for exactly two seconds, but function calls and other factors could have extended the relay on time. I had to adjust my counter variable and modulus counter to get as close as I could to three seconds.

For sure, the project worked because I had ADC checking built into my code which would allow the relay to turn on. In the lab I demonstrated my project to the TA, by showing how when motion is detected the light will turn on and continue to stay on for approximately three seconds. Continually he tried to turn on the light with motion, but the hardware has potentiometers on them with can set a delay or adjust the sensitivity. The delay potentiometer was increased a little, so users can't turn on the light with motion immediately once the light has been turned on and off. The TA checked me off once I told him what I created, the functions of it, and after he tried it himself.

What I'd do differently next time?

To be honest I wish I would have added a different feature or more features to the LED lamp. My project was based on my budget and I couldn't fit into my budget more parts for the project. Life as a college student is hard, and I truly wish I could have done more.

Possible features I would have added would be an LCD screen to state when the light is on and time until the light turns off. Personally, I really would've enjoyed using an LCD in my project. The LCD screen is already included in the lab kit, which is nice, it's just the idea to use the LCD didn't come to mind until later in the semester.

Project Photos:

Figure 5.1: Front of LED Lamp. Shows the PIR sensor built into the lamp along with battery terminal and breadboard circuit. Breadboard circuit contains this courses microcontroller (MC9S08), decoupling capacitor, and a 5V and 3.3V regulator.

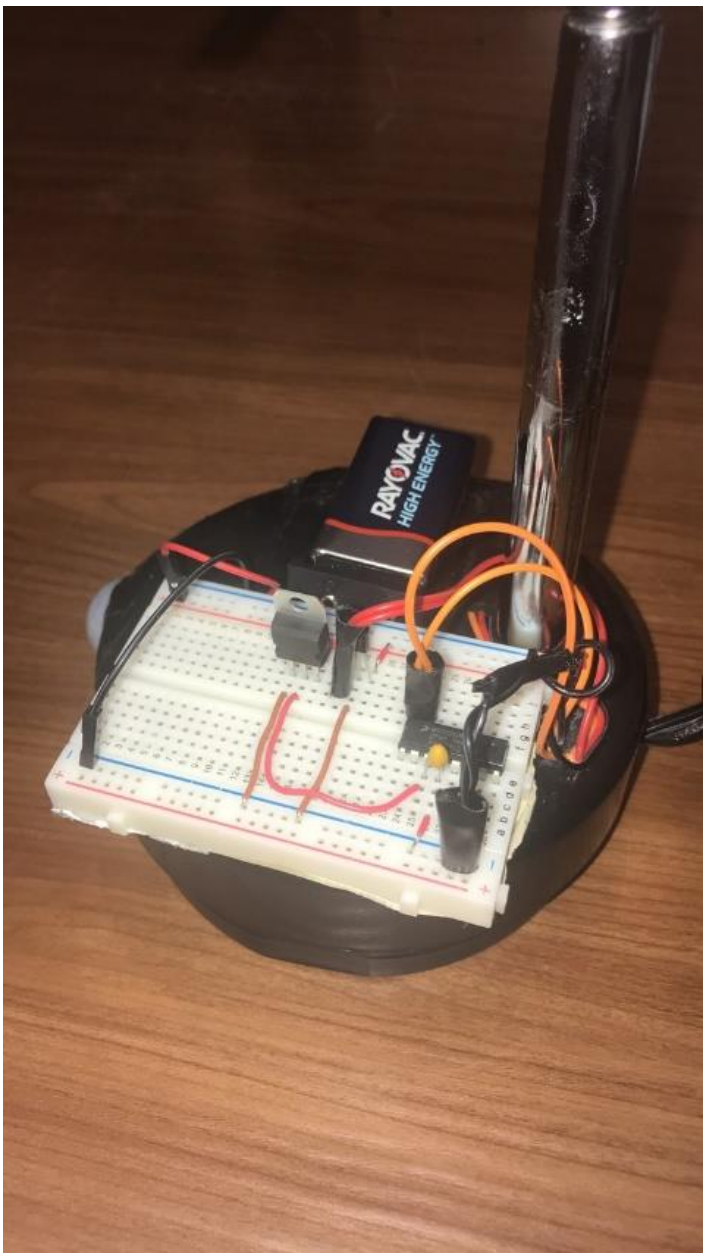


Figure 5.2: Same as figure 1 but from a different angle. Shows breadboard circuit primarily. Contains all components listed in figure 1.

Figure 5.3: LED lamp innards. Contains 5V relay (Normally closed setup), PIR sensor, and wiring for every component. I wanted to make most of the component fit in with the chassis of the LED lamp.

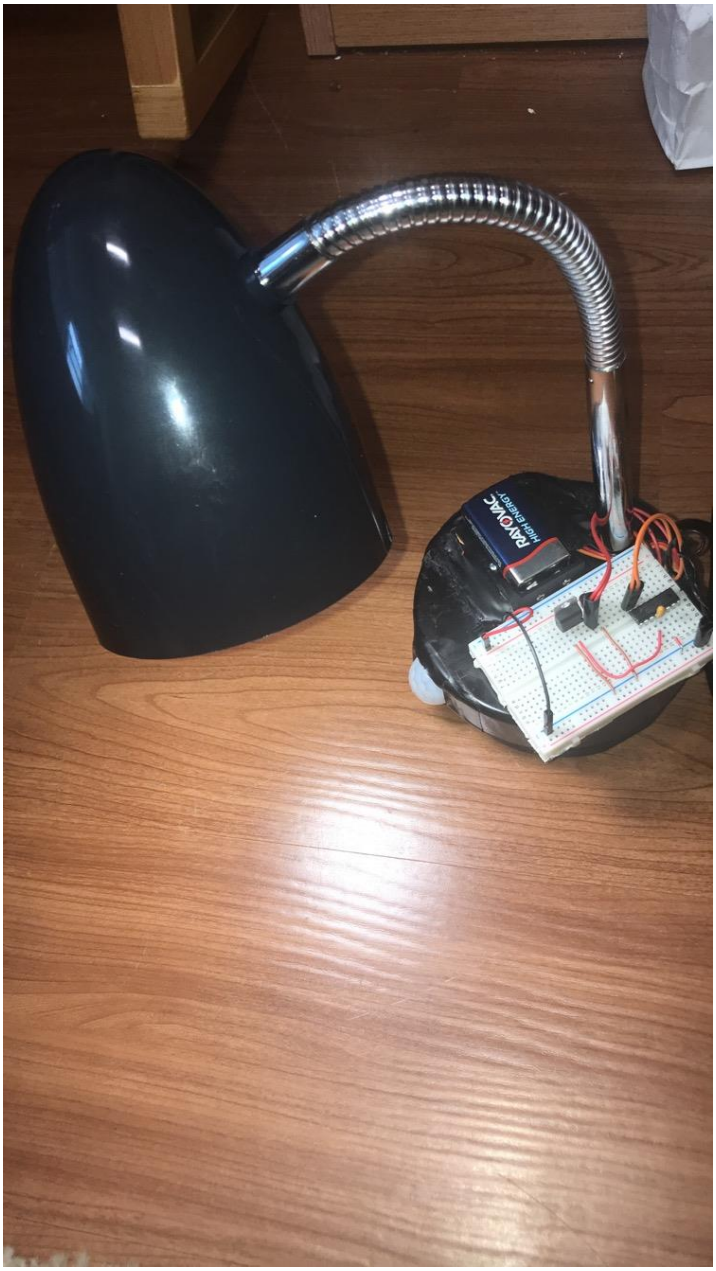


Figure 5.4: Contains everything listed in figure 1, but shows the lamp at a larger POV

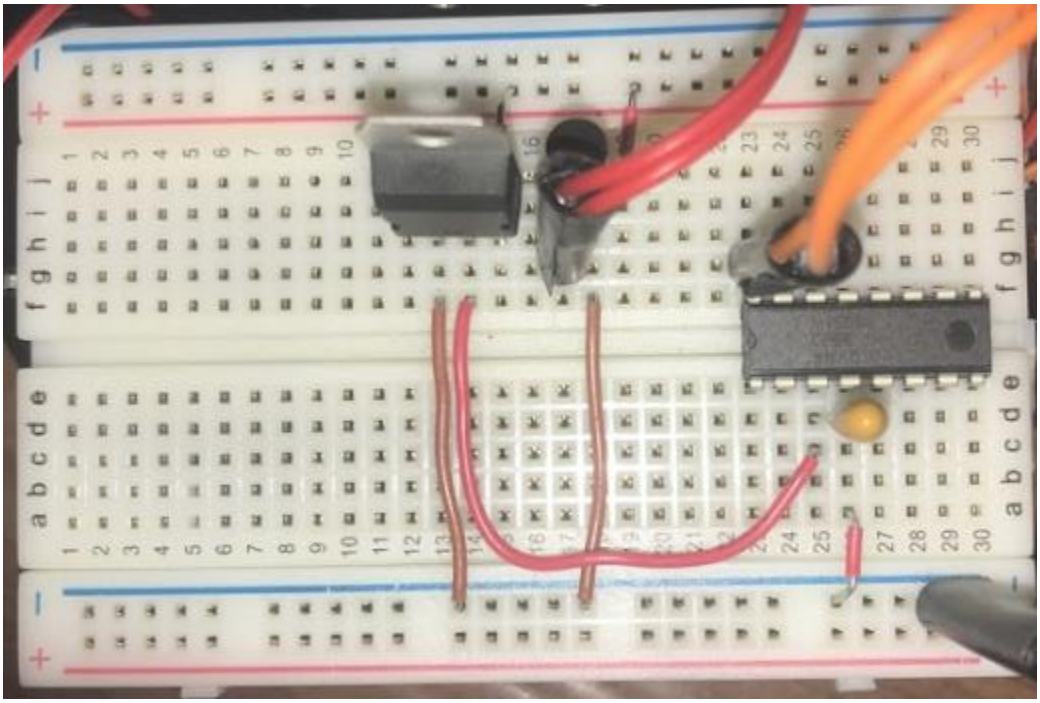


Figure 5.5: Up close shot of my project breadboard setup.

Contains: Microcontroller (MC9S08QG8), decoupling capacitor, a 5V and 3.3V linear regulator, and wires.

Appendix

Appendix 1 What I Did?

1.1 (Paragraph 1) - How I came up with my project Idea.
1.2 (Paragraph 2) – Problem I tried to solve.
1.3 (Paragraph 3) – Why commercial solution didn't appeal to me.

Appendix 2 How I Did it?

2.1 (Paragraph 1) – How I went about creating my hardware setup and code.
Figure 1.1 (After Paragraph 1) – Diagram showing hardware setup.
2.2 (Paragraph 2) – Problems I ran into when creating my project.

Appendix 3 How I Know It Worked?

3.1 (Paragraph 1) – How I knew my project worked.
3.2 (Paragraph 2) – Teachers Assistant demonstrations.

Appendix 4 What I'd do different next time?

4.1 (Paragraph 1) – Reasoning behind not doing more features in project.
4.2 (Paragraph 2) – Project idea I'd do in place of my current project if I were to restart.

Appendix 5 Project Photos

Figure 5.1-5.5 - Shows the hardware setup inside the lamp and on breadboard.

Appendix 6 Project Code

Contains all code add motion detection to an LED. The lamp is based upon an MTIM interrupt.

```
#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */
#define lowcutoff  153 /* ~3.0V (2.988V) 8-bit converted reading. */
#define highcutoff 256 /* ~3.333V (3.339V) 8-bit converted reading. */

// Function prototypes here (any function that is CALLED)
```

```
void cpuinit(void);
void mtiminit(void);
void pininit(void);
void flaginit(void);
void TurnOnLight(void);
void TurnOffLight(void);
void TOFdisable (void);
void TOFenable (void);
void SetCounter(void);
void resetCoco(void);

// Global variable declarations go here
unsigned int counter;
unsigned int analogRead;
int LightOffFlag;
void main(void)
{
  // Variable declarations go here

  counter = 248; //Counter for 2 seconds of on time

  // Initialization routines called here
  cpuinit();
  pininit();
  mtiminit();

  EnableInterrupts;

  // Main loop
  for(;;)
  {

  }
  /* loop forever */

  /* never leave main */

}
void cpuinit(void)
{   /* Turn off watchdog and trim oscillator */
          SOPT1 = 0x53;
          ICSTRM = NVICSTRM;
}

/* Sets up MTimer specificaitons. */
void mtiminit(void)
{
          // Enables MTIM overflow interrupt.
          MTIMSC_TOIE = 1;
          MTIMSC_TSTP = 0;
```

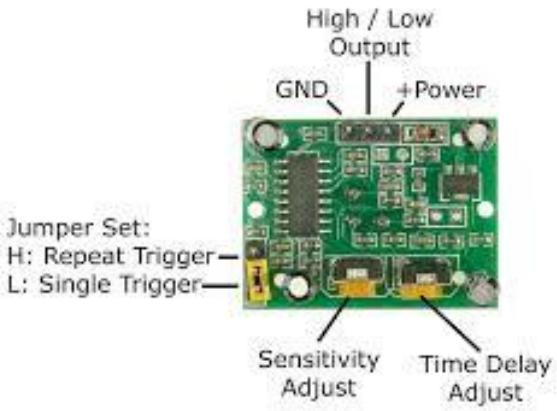
```
        // MTIM clock uses bus clock.
        MTIMCLK_CLKS = 0;

        // Clock source prescaler  is 256
        MTIMCLK_PS = 8;

        // Modulus for 2 seconds of PTAD_PTAD1 high time.
        MTIMMOD = 248;
}

/* Set up all pins, including the ADC pins. */
void pininit(void)
{
  // Initializes PTAD direction and data..
  PTADD_PTADD1 = 1;
  PTAD_PTAD1 = 0;

  // Sets input channel to 0.
  ADCSC1_ADCH = 0;

  // Initiates conversion via software.
  ADCSC2_ADTRG = 0;

  // Disables I/O control for ADP0 (pin 16).
  APCTL1_ADPC0= 1;

  // Sets mode to 8- bit conversion.
  ADCCFG_MODE = 0;

}

/* Light off flag to prevent ADC conversions */
void flaginit (void)
{
   LightOffFlag == 1;
}

/* Turns relay on */
void TurnOnLight(void)
{
        PTAD_PTAD1 = 1;
}

/* Turns relay off */
void TurnOffLight(void)
{
        PTAD_PTAD1 = 0;
}

/* TOF flag set to 0 */
void TOFdisable (void)
{
```

```
        MTIMSC_TOF = 0;
}

/* TOF flag set to 1 */
void TOFenable (void)
{
        MTIMSC_TOF = 1;
}

/* Sets counter to match MTIMMOD for 2 sec interrupt */
void SetCounter(void)
{
        counter = 248;
}

/* Sets Coco flag to low */
void resetCoco(void)
{
  ADCSC1_COCO = 0;
}

/* Interrupt occurs once every 8 ms */
interrupt VectorNumber_Vmtim void mtim_tof_isr(void)
{
  MTIMSC;

  // Clears TOF flag to low.
  TOFdisable();

  // Conversion with prevention check.
  if (ADCSC1_COCO && LightOffFlag == 1)
  {
     analogRead = ADCRL;
  }
  // Checks to see if PIR sensor detected anything post conversion.
  if (analogRead >= lowcutoff && analogRead <= highcutoff)
  {
      LightOffFlag = 0;
      TurnOnLight();
  }
  // Decrements counter.
  if(PTAD_PTAD1 == 1 && counter > 0)
  {
    counter--;
  }

  else
  {
    // Sets TOF Flag high
    TOFenable();

    // Turns relay off.
```

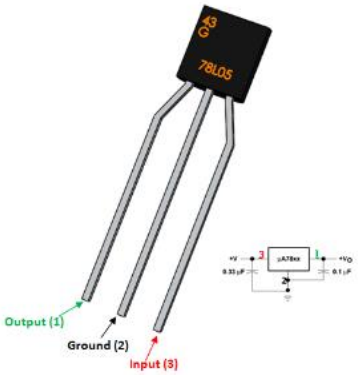
```
            TurnOffLight();

            // Resets counter to initial value.
            SetCounter();

            // Lowers Conversion complete flag.
            resetCoco();

      // Sets light on flag to starting state.
            flaginit();
    }
}
```

Appendix 7 Schematics and Diagrams

Diagrams for my hardware used.



HC-SR501 Analog PIR Motion Sensor Diagram



78L05 5V 100mA Linear Regulator

LD33V 3.3V 800mA Linear Regulator

5V Relay Terminals and Pins

SRD-05VDC-SL-C 5V relay