ECE 376 - Homework #5

Stepper Motors, NeoPixels, Analog Inputs.

Egg Timer with a Stepper Motor

1) Requirements:

Inputs

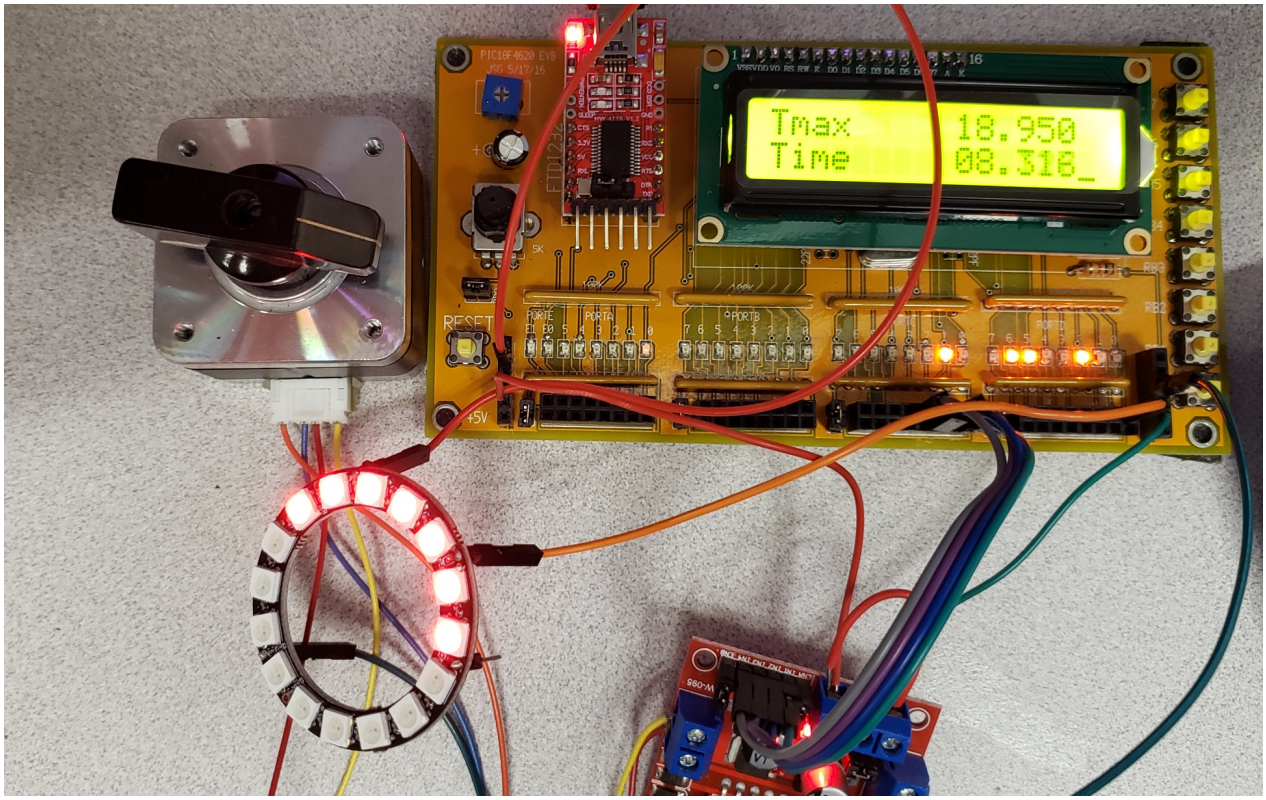
- Push button RB1 (start the game)
- Analog input (RA0)

Outputs

- Stepper Motor (PORTA)
- NeoPixel (RD0 - part 2)
- LCD display (PORTD)

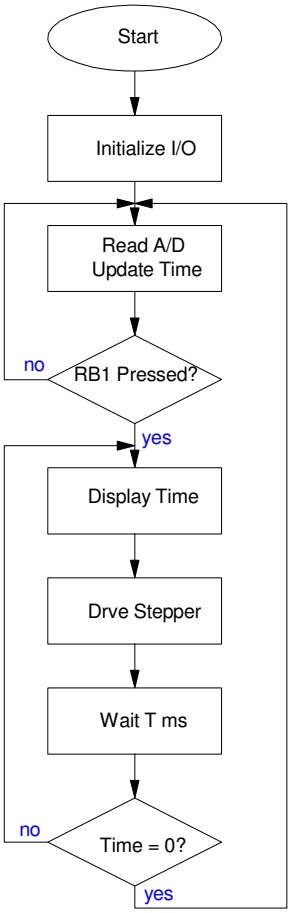
Relationship

- Turn the A/D input to set the time T from 0.0 to 50.0 seconds
- Press RB1 to start the egg time
- The stepper motor then turns to 180 degrees (100 steps) in T seconds
- The LED displays 0% to 100% of the lights as time goes from 0% to 100% (part 2)
- It then waits 1 second then returns to 0 degrees (0 steps)
- Repeat



2) C code, flow chart, and resulting number of lines of assembler

```
while(1) {

   TIME = 0;
   while(!RB1) {
      TIME = A2D_Read(0)*50;
      LCD_Move(0,8);  LCD_Out(TIME, 5, 3);
   }


  dt = TIME/100;
  ms = 0;
  STEP = 0;
  while(ms < TIME) {
     STEP += 1;
     PORTC = TABLE[STEP % 4];
     // BarGraph(STEP);
     LCD_Move(1,8);  LCD_Out(ms, 5, 3);
     Wait_ms(dt);
     ms = ms + dt;
     }

  Wait_ms(1000);

  while(STEP > 0) {
      STEP -= 1;
      PORTC = TABLE[STEP % 4];
      // BarGraph(STEP);
      Wait_ms(10);
      }

  }
}
```



Compilation Restuls

```
Memory Summary:
    Program space        used   CCCh (  3276) of 10000h bytes   (  5.0%)
    Data space           used    33h (    51) of   F80h bytes   (  1.3%)
    EEPROM space         used     0h (     0) of   400h bytes   (  0.0%)
    ID Location space    used     0h (     0) of     8h nibbles (  0.0%)
    Configuration bits   used     0h (     0) of     7h words   (  0.0%)
```

3) Validation: Collect data in lab to verify you met the requirements.

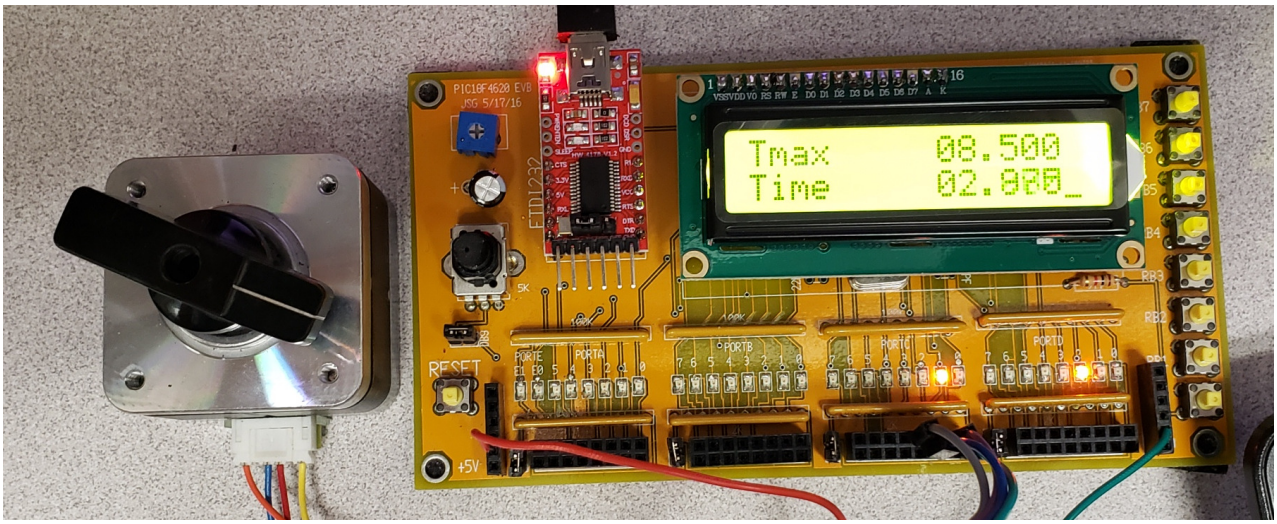
The analog input adjusts the time from 0.0 to 50.0 seconds

- check

Pressing RB0 starts the egg timer

- check
- 10.85 seconds actually takes 11.69 seconds
- 20.35 seconds actually takes 21.59 seconds
- The stepper motor turns 180 degrees in T seconds (about) then turns back to zero

4) Demo. Video or in person.



Egg Timer with NeoPixels

5) Requirements: Specify the inputs / outputs / how they relate.

Add a NeoPixel

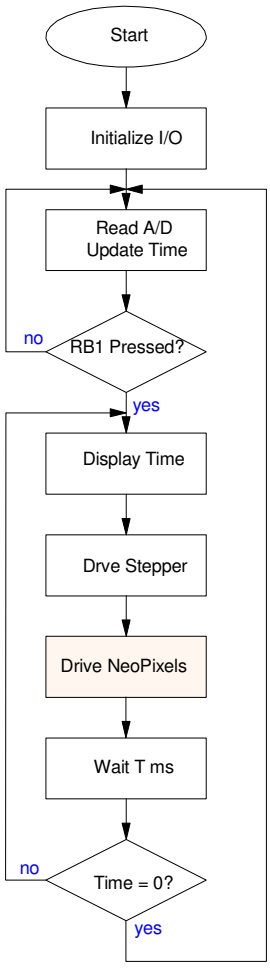
- Turn on 0 to 16 NeoPixels
  - 0% = no lights on
  - 100% = all 16 lights on
  - Proportional inbetween

6) C code, flow chart, and resulting number of lines of assembler

Add a subroutine which drives the NeoPixels

- Call it as you update the stepper motor position

```
void BarGraph(unsigned char pct) {
   unsigned char RED[16], i, N;
   TRISD0 = 0;
   N = 0.16*pct;
   for(i=0; i<16; i++)
      if(i < N) RED[i] = 50;
      else RED[i] = 0;

   NeoPixel_Display(RED[0],0,0);
   NeoPixel_Display(RED[1],0,0);
   NeoPixel_Display(RED[2],0,0);
   NeoPixel_Display(RED[3],0,0);
   NeoPixel_Display(RED[4],0,0);
   NeoPixel_Display(RED[5],0,0);
   NeoPixel_Display(RED[6],0,0);
   NeoPixel_Display(RED[7],0,0);
   NeoPixel_Display(RED[8],0,0);
   NeoPixel_Display(RED[9],0,0);
   NeoPixel_Display(RED[10],0,0);
   NeoPixel_Display(RED[11],0,0);
   NeoPixel_Display(RED[12],0,0);
   NeoPixel_Display(RED[13],0,0);
   NeoPixel_Display(RED[14],0,0);
   NeoPixel_Display(RED[15],0,0);
   }
```



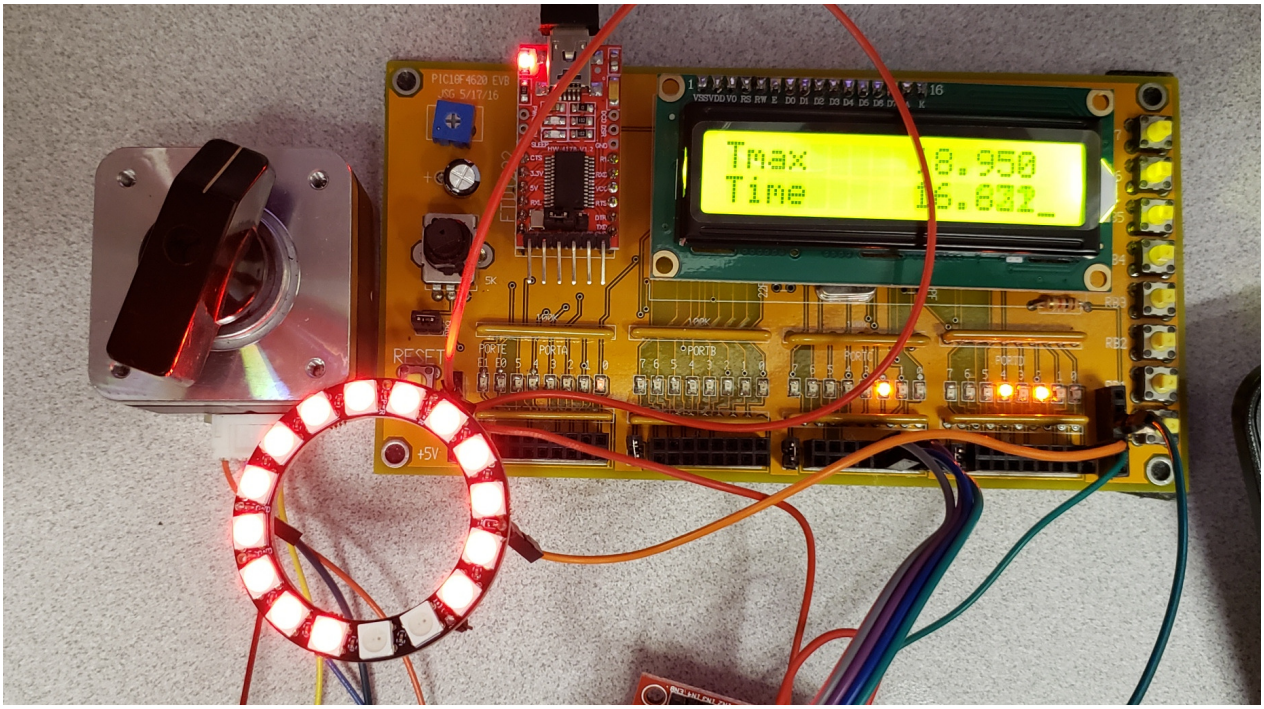
Compilation Results

```
Memory Summary:
    Program space        used  15D4h (  5588) of 10000h bytes   (  8.5%)
    Data space           used    3Dh (    61) of   F80h bytes   (  1.5%)
    EEPROM space         used     0h (     0) of   400h bytes   (  0.0%)
    ID Location space    used     0h (     0) of     8h nibbles (  0.0%)
    Configuration bits   used     0h (     0) of     7h words   (  0.0%)
```

7) Validation: Collect data in lab to verify you met the requirements.

The NeoPixel turns on lights with how much of the cycle is complete

- 0% = no lights
- 50% = 1/2 of the lights
- 100% = all of the lights



8) Demo. Video or in person.