5-2018

Automated Farm Tunnels: Scaled Prototype

Axel Anderson
Trinity University, aanders2@trinity.edu

Bryan Granizo
Trinity University, bgranizo@trinity.edu

Samuel Studebaker
Trinity University, sstudeba@trinity.edu

Automated Farm Tunnels: Scaled Prototype

Group E: Axel Anderson, Bryan Granizo, Samuel Studebaker

May 3, 2018

Table of Contents:

Design Summary

The use of covered tunnels in agriculture has increased dramatically in the Texas Hill Country over the past few years. These tunnels are similar to greenhouses but do not have climate control so they need natural venting for humidity and temperature purposes, which means that the plastic covering the tunnel (see Figure 1) needs to be raised and lowered repeatedly. The first generation of these tunnels seen in Figure 1 did not have a mechanism to roll the plastic up or down, but the second generation tunnel shown in Figure 2 has implemented a pipe, which will be referred to as the "plastic rolling pipe", that goes down the length of the tunnel and is turned by a gear box to raise or lower the sides. The plastic rolls around the pipe and pulls the pipe up the tunnel or unravels and allows the pipe to descend.

Figure 1. Haygrove® High Tunnels in use at Studebaker Farms

Plastic rolling pipe
(partially rolled down)

Figure 2. Next generation Haygrove® tunnels used at Studebaker Farms.

We wanted to build a scaled down prototype to simulate automated control of two tunnel motors with user defined temperature setpoints and manual override. The device contains two DC motors in an automated loop; the loop is controlled by a user input for temperature to turn the motors in one direction, and another temperature input to turn the motors in the other direction. The two motor directions are to simulate the raising and lowering of the tunnels. Two different setpoints are inputted because it is desired to lower the tunnels at a warmer temperature to trap heat and to raise the tunnels at a lower temperature to allow ventilation to occur earlier. Having two different temperatures, however, creates some dilemmas. What happens if the temperature exceeds the temperature to raise the tunnel but never reaches the temperature to lower? To solve this problem, our device analyzes the change in temperature and raises the tunnel if the temperature is increasing between the two setpoints but lowers the tunnel if the temperature is decreasing between the two setpoints. For example, if a user inputs a lowering temperature of 30 °C and a raising temperature of 20 °C, the device will lower the tunnel if the temperature is in between 20 and 30 °C and the temperature is decreasing; this means the weather is getting colder and the tunnel should close immediately to trap heat. The device compares two temperature measurements taken with a significant amount of time between them, which prevents the motors from turning on rapidly due to small fluctuations in temperature. For the purpose of the prototype demonstration, the temperatures were taken about 10 seconds apart, but the time would be increased to several minutes in actual implementation.

The device also allows the user to interrupt the loop and remove a motor from the loop. The user can then set the removed tunnel to either stay up or down. Both motors can be removed from the loop and added to the loop. When a “tunnel” is added to the loop and its current state does not match the current state of the loop, the tunnel will raise or lower to match the loop. Furthermore, the device will not try to raise or lower the tunnel if the tunnel is already in that position. For the purpose of this prototype, the motors ran for a certain time period, but this would be changed for actual implementation on a tunnel. We would have two limit switches on the tunnels that would allow the device to know if the tunnel was raised or lowered. Our goal for the project was to create a scaled down prototype that demonstrated the desired motor control. A prototype to actually perform the control on tunnels would be expensive- each motor alone would cost $100+. However, the group did design what a full sized prototype would look like. This design includes antennas for transmitting and receiving RF signals, which were not successfully implemented in the project.

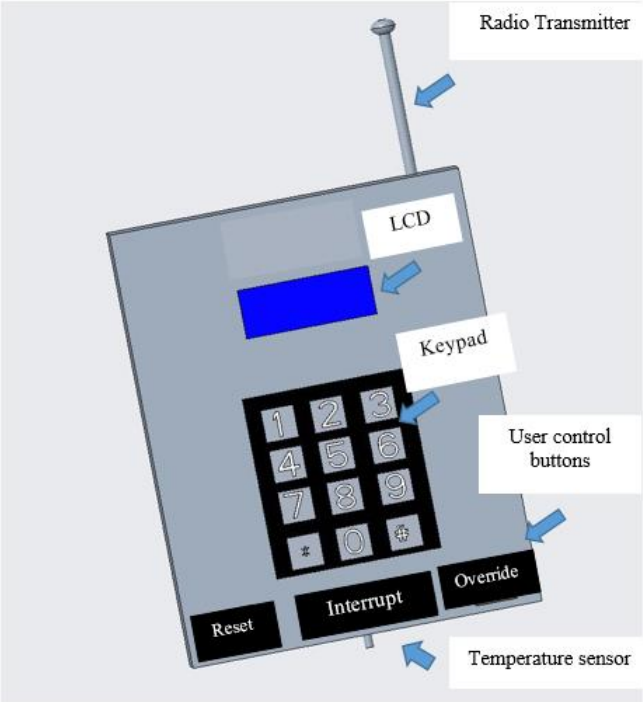


Figure 4. Main control box with user interface

The main control box seen in Figure 4 would communicate with other smaller control boxes connected to each motor. Each motor would be adapted to the current tunnel system described earlier as shown in Figure 5. The device was not consolidated into a control box, but the desired function of the device was achieved by the group.

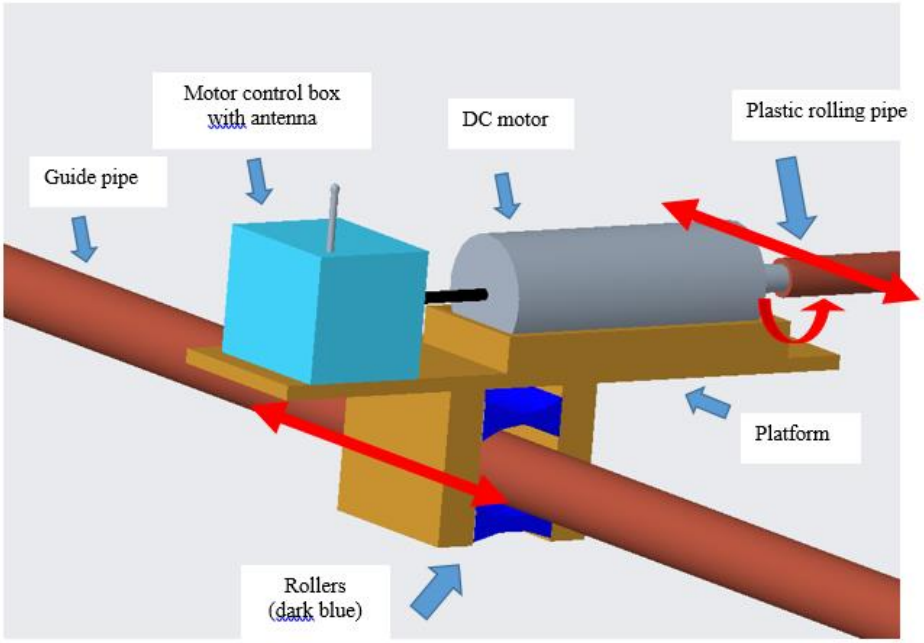


Figure 5. Motor configuration with tunnel pipes, red arrows show movement of configuration

The project successfully implemented the desired automated temperature control and manual override and is a good foundation for implementing a similar system on a larger scale tunnel.

System Details

The system's user interface consisted of a keypad, three push buttons, and a liquid crystal display (LCD). The keypad was used to input the temperature setpoints and for override control. The "interrupt" button was used to interrupt the automated loop, which allows the user to press one of the other two buttons to either override the tunnel motors or to reset the temperature setpoints. The LCD was used to display the current temperature reading, the temperature setpoints, the change in temperature, and instructions for manual override. As seen in the functional diagram below, a microcontroller used inputs from the user interface and a temperature sensor to control the LCD, LEDs, and two motor-speaker pairs. The controller was used to control LEDs to visualize the current state (up or down) of the motors and of the automated loop. For example if the LED is on for motor 1, it signifies that tunnel 1 has been raised. A LED was also used for the automated loop to show the current state of the loop, which allowed the user to see the behavior of the loop even if both tunnels had been manually removed. The LEDs were used since the prototype had stationary motors that spun but did not turn an external load that allowed the users to tell when the "tunnel" was in a certain state.

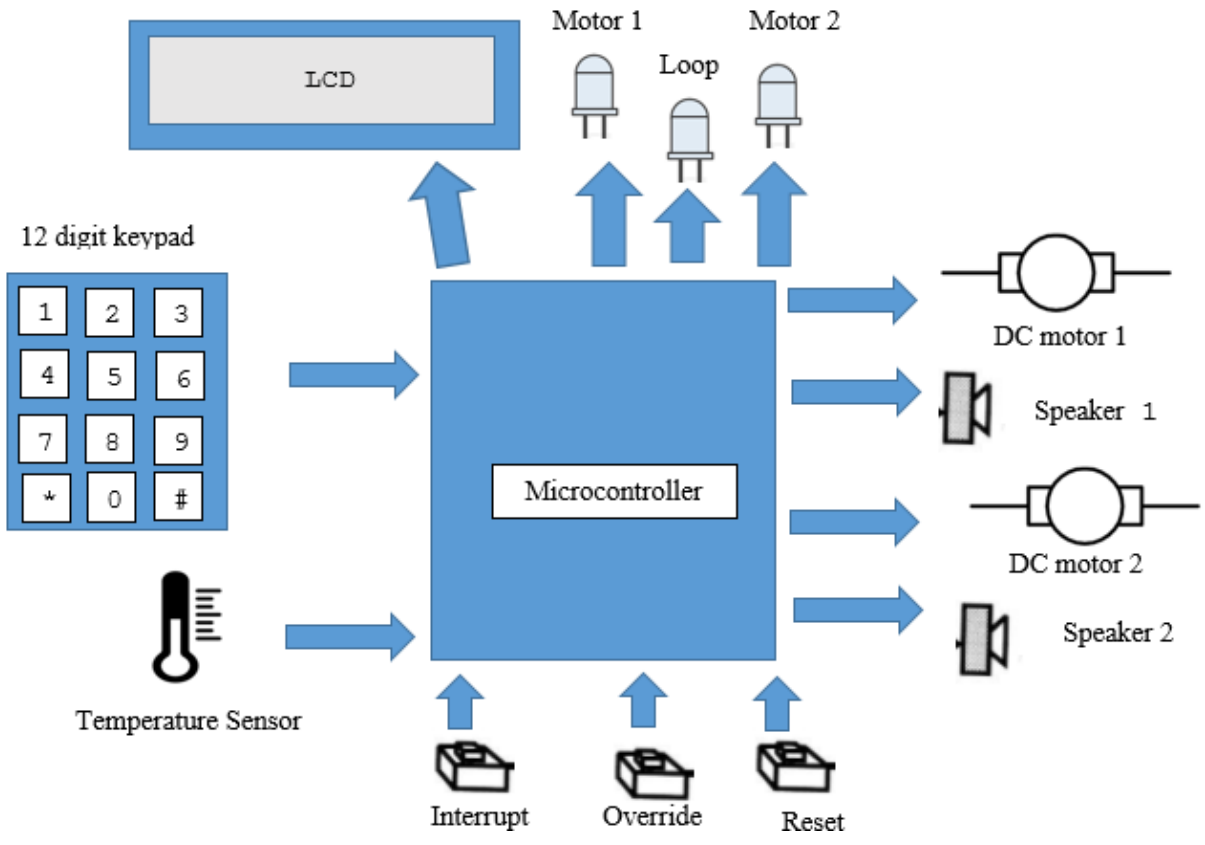


Figure 3. System functional diagram

The input and output requirements for the system are shown in Table 1. The motors were controlled using an H-bridge, which needed inputs for the motor speed and for directional control. Speakers were connected to the motors and were turned on 1 second before the motor started turning to serve as a warning that the motors would be turning on. The H-bridge used a 9 V battery to power the motors. The keypad used a polling method that controlled the columns (outputs) and read the rows as inputs. The total amount of I/O connections for the microcontroller was 26, so the group chose the largest microcontroller available, the PIC 16F1937, for the project.

Table 1. I/0 Requirements

Component	Data type	I/O
LCD	Digital	Output (6)
Push buttons	Digital	Input (3)
Temperature sensor	Analog	Input (1)
Keypad	Digital	Output (3), Input (4)
Motors	Digital	Output (6)
Speakers	Digital	Output (0*)
LEDs	Digital	Output (3)

*Speakers were connected to the output to a motor and did not require their own output

The device used a TMP36GSZ temp sensor, which outputs a voltage based on the temperature that needed an analog to digital conversion. The sensor was calibrated based on room temperature measurements, and the digital bin to temperature conversion in Celsius was achieved using Eq. 1.

$$T = (\frac{Digital_{bin}}{1023} \cdot 5\,V \cdot 1000 - 475)/10 \quad (1)$$

Implementing Eq. 1 into the code was a little more complicated since we could not get the compiler to keep decimal points (floats). We got around this using clever multiplication and the modulus function. An overview of the code is provided in the following figures. The flowcharts in Figure 6 cover the function of the automated loop, and the second flowchart in Figure 7 covers the interrupt code. The full code is provided in Appendix B. The pin and port assignments for the components are shown in Table 2. The full wiring diagram for the device is in Appendix B.

Table 2. PIC Port assignments

Component	Component Pin	PIC Port
LCD	D7	PORTA.3
LCD	D6	PORTA.2
LCD	D5	PORTA.1
LCD	D4	PORTA.0
LCD	E (enable)	PORTA.5
LCD	RS (register select)	PORTA.4
Temp sensor	Voltage output	PORTE.0 (AN5)
LED-loop	NA	PORTD.2
LED-motor 1	NA	PORTC.0
LED-motor 2	NA	PORTC.1
Interrupt	Resistor-gnd	PORTB.0
Override	Resistor-gnd	PORTD.1
Reset	Resistor-gnd	PORTD.0
Keypad	Row 1 (R1)	PORTB.4
Keypad	Row 2 (R2)	PORTB.5
Keypad	Row 3 (R3)	PORTB.6
Keypad	Row 4 (R4)	PORTB.7
Keypad	Column 1 (C1)	PORTB.3
Keypad	Column 2 (C2)	PORTB.2
Keypad	Column 3 (C3)	PORTB.1
H-bridge	1,2EN (motor 1 enable)	PORTC.7
H-bridge	1A (motor 1 directional control)	PORTC.6
H-bridge	2A (motor 1 directional control)	PORTC.4
H-bridge	4A (motor 2 directional control)	PORTD.6
H-bridge	3A (motor 2 directional control)	PORTD.5
H-bridge	3,4EN (motor 2 enable)	PORTD.7

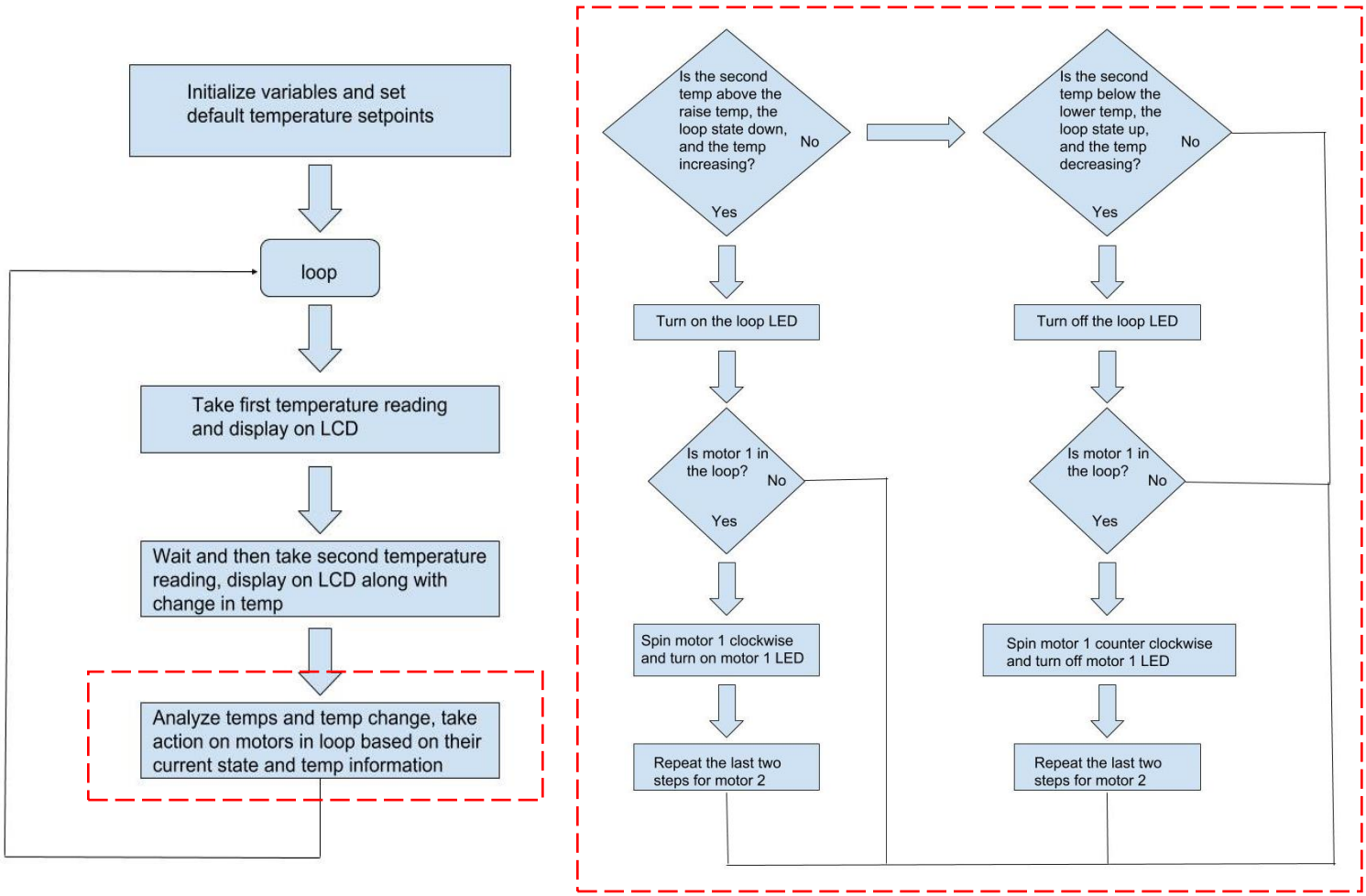


Figure 6. Flowchart for overview of automated loop (left), and motor control inside of loop (right)

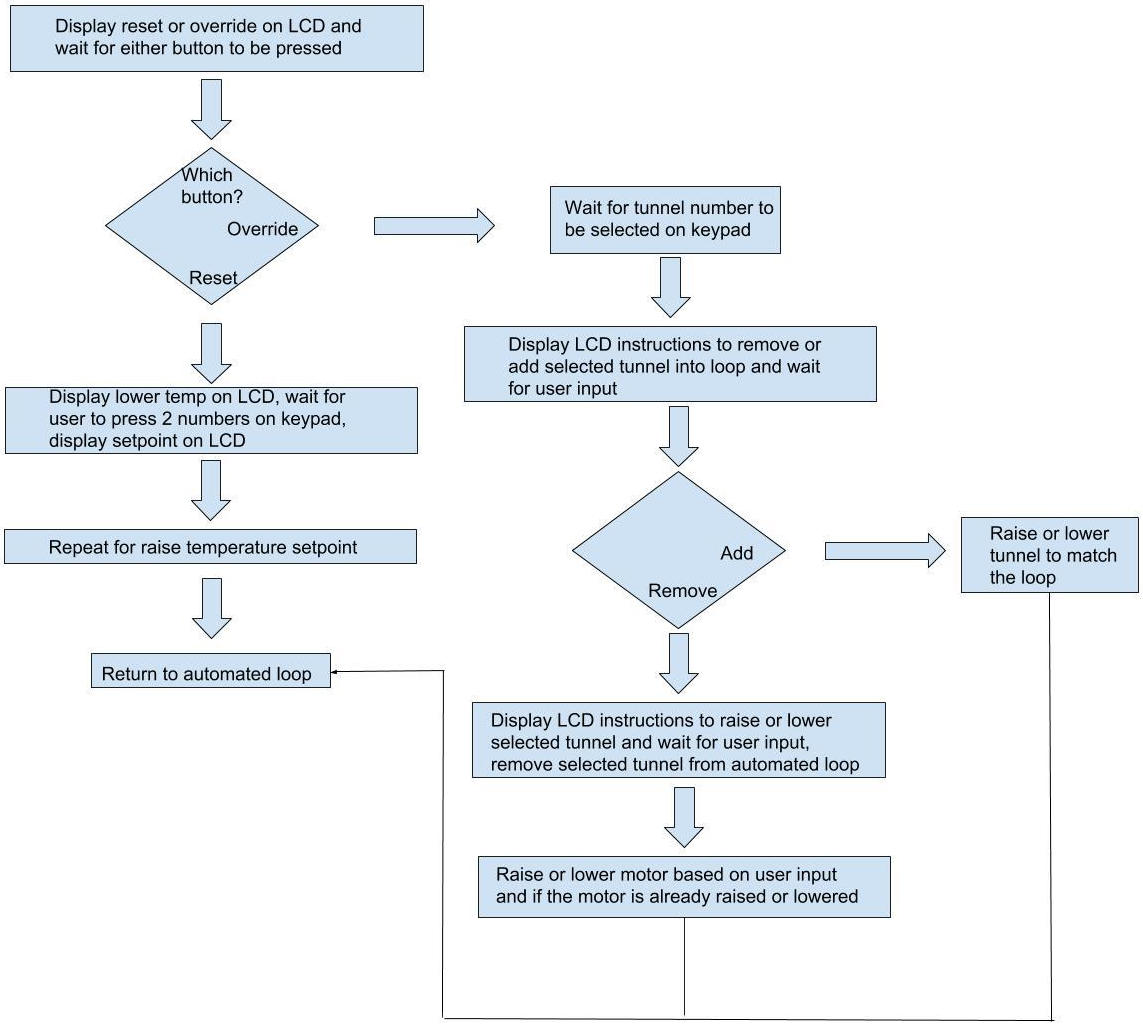


Figure 7. Flowchart for interrupt

Design Evaluation

Output Display

The system output display included LEDs and a LCD. The book included a small section on a LCD but did not discuss in detail how to set up the LCD and what the command "lcdout" did. We had to do some research to understand that lcdout initializes LCD pins to certain ports in the PIC. Since some ports in the PIC 16F1937 had specialized functions, we could not use all of the predefined lcdout initialized pins. Thus, we had to do some research to find out how to change the designated ports in lcdout. We believe this is a rating of 15 because we had to do some research, but the text book does discuss LCDs and there was adequate documentation on the PICBasic Pro website. The LCD functioned

as designed and experienced no problems during the demonstration. The LEDs were also fully functional but did not require any research.

Audio Output Device

The system used a buzzer, which did not require much research. The buzzer was run in parallel to motor control and had virtually no interface circuit. We believe this is a rating of 10 because the buzzers worked as expected and did not require much effort or research.

Manual User Input

The system used push buttons and a keypad for manual user input. We used a keypad in lab and that part of the project code was based on the lab code. We did, however, use the keypad slightly different and had to adjust the code to work properly for our project. The lab code continuously polled the keypad, and the project needed to only poll at a given time, needed to store more than one number, and needed to wait for a certain number of keys to be pressed before continuing. This required some coding effort, but the interface circuit and coding set up was used from the lab. We already knew how to interface push buttons, so those did not require much effort. We also implemented one button as an interrupt. We discussed this in class, but it required research to configure an interrupt for our PIC and to get it to work within our code. We believe this is a rating of 15.

Automatic Sensor

The system used a temperature sensor that produced an output voltage based on the temperature. Thus, we needed to perform an analog to digital conversion of the voltage similar to one we did in lab, but we had to convert that value to a temperature. We had to do research on the temperature sensor to understand the voltage-temperature curve and we to convert the digital value to a temperature value for the LCD display. We then used room temperature measurements to adjust the code to calibrate the sensor. This required effort and research, and we believe it is a rating of 15.

Actuators, Mechanisms & Hardware

The system used two reversible DC motors that used SN5441ONE H-bridges, which allowed us to turn the motors in both directions. We had to do research to understand how H-bridges worked and how to interface them with our PIC. We successfully turned both motors in each direction and believe this is a rating of 15.

Logic, Processing, and Control; And Miscellaneous

The system used an extensive amount of logic and had some menu-driven software. The code had to convert the analog voltages to temperatures and compare different measurements. The code stored a previous temperature measurement to be able to analyze the change in temperature. We also had to implement an interrupt and adjust the code based on our research into how the interrupt works and how to configure it for our PIC. The code also had a menu-drive user interface once the interrupt was activated. This code had to add and remove motors from the automated loop and allowed for each motor to be individually controlled. The code took a lot of effort and research, so we believe this is a rating of 20.

Partial Parts List

H-Bridge: SN5441ONE, Digikey, $2.52

Temp sensor: TMP36GSZ, Digikey, $1.49

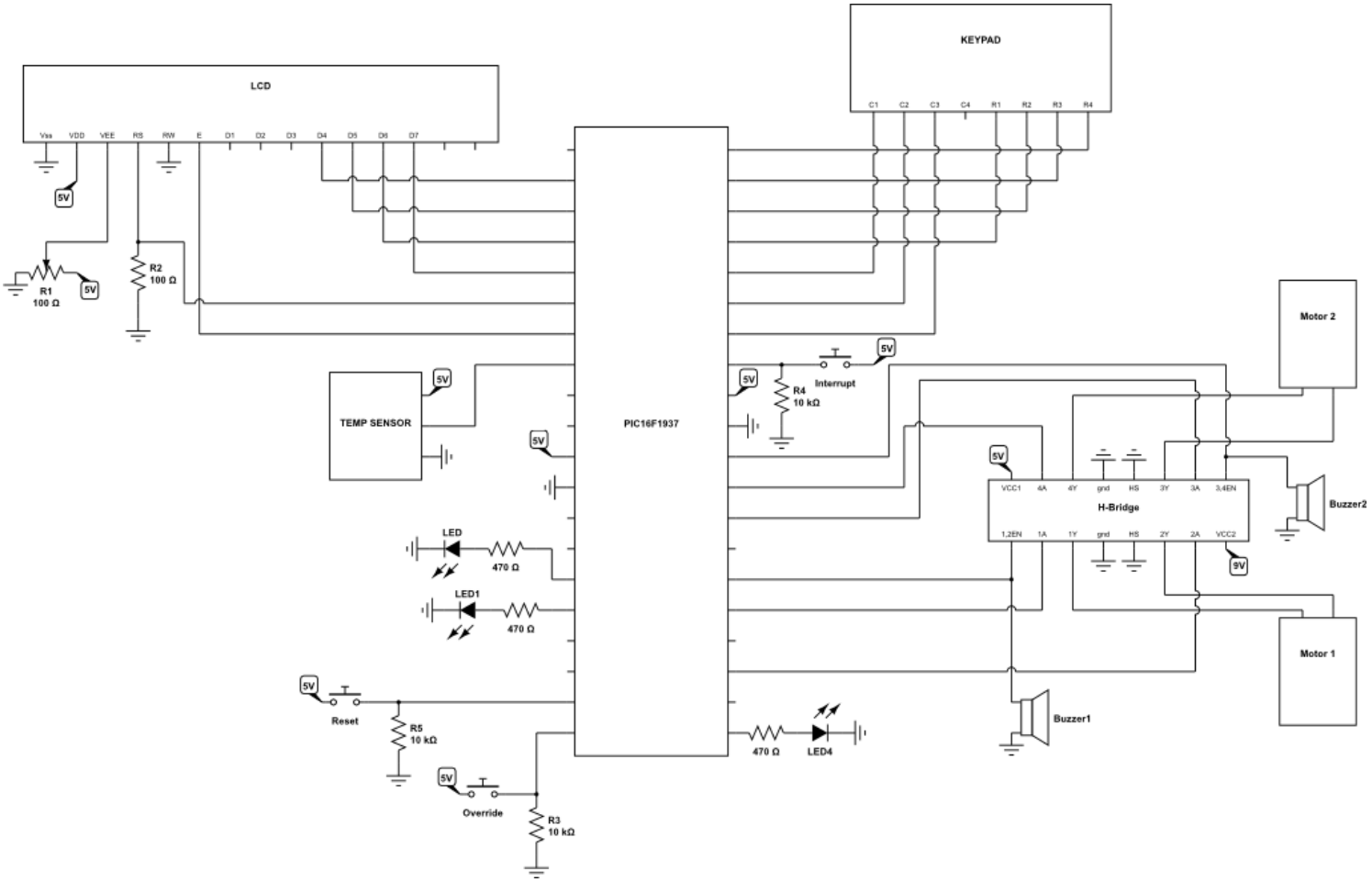
RF module: RFM69HCW, Digikey, $4.74

Lessons Learned

Our first lesson learned was that using our selected RF module was very difficult with a PIC. The manufacturer had a set up guide but it was for an Arduino and had an example code that used an already built in Arduino library. We tried to use that as reference but were unable to get it to work with a PIC. To incorporate RF communication, we would recommend using an Arduino. Our next lesson was using an interrupt in the compiler. It turns out that the interrupt will register but will wait until the command it occurs during will finish. For us, we had a long pause between temperature measurements, which meant the interrupt wouldn't affect the system for a long period of time. We wanted the interrupt to be immediate so that the user didn't have to wait. We accomplished this by breaking up the pause into a series of shorter pauses. Another way this could have been accomplished is to have the automated loop run by another controller that communicates with the user interface run by another controller. An Arduino could have been implemented for one of these to allow for wireless communication and for the user interface to run parallel to the automated loop. Running it in parallel could allow the user to change the setpoints or override a tunnel and then interrupt the automated loop with that information.

Appendix

Appendix A: wiring diagram



Appendix B: full code

```
 'code to poll keypad adapted from book:
Introduction to Mechatronics and Measurement
Systems 4th edition, Alciatore and Histand,
McGraw Hill, page 300-301
'*************************************************************    '*
'Name   : Automated 2 motor control with manual override   *   '*
'Author  : Samuel Studebaker                                *
'*  Notice  : Copyright (c) 2018 [select VIEW...EDITOR OPTIONS] *
'*          : All Rights Reserved                           *
'*  Date    : 4/30/2018                                     *
'*  Version : 1.0                                           *
'*  Notes   :                                               *
'*          :                                               *
'*************************************************************
 on interrupt goto myint    'set up for interrupt
'internal oscillator setup
define OSC 8           '8 MHz
OSCCON.3 = 0
OSCCON.4 = 1
OSCCON.5 = 1
OSCCON.6 = 1
'ADC setup for temp sensor
adcVar var word
voltage var word
temp var word
xx var word
yy var word
tt var word
xxx var word
yyy var word
ttt var word
temp1 var word
temp2 var word
temp3 var word
temp4 var word
define ADC_BITS 10
define ADC_CLOCK 3
define ADC_Sampleus 15
TRISE=%11111111    'set port E to be inputs
ANSELE.0=1         'turn on ansel register for port e.0
ADCON0=%00010111   'activate analog channel AN5 (port e.0), enable ADC
ADCON1.7=1         'set A/D to right justified
'LCD setup
TRISA=%00000000     'set port A to be outputs
ANSELA=%00000000     'turn off A/D conversion for port A
define   LCD_EREG PORTA   'reset lcdout to have the enable at porta.5
define LCD_EBIT 5
'keypad setup and other internal variables
  ' Enable PORTB pull-ups
  OPTION_REG = $7f
  'set appropriate input/outputs for keypad, rows are inputs, columns outputs
  TRISB = %11110001
  row1 var PORTB.4
  row2 var PORTB.5
  row3 var PORTB.6
  row4 var PORTB.7
  col1 var PORTB.3
  col2 var PORTB.2
  col3 var PORTB.1
  rownumber var byte
  keycol1 var byte
  keycol2 var byte
  keycol3 var byte
```

```
  keycol4 var byte
  numblink var byte
  numblinks var byte
  numblink3 var byte
  numblink4 var byte
  key var byte
  S var byte
  T var byte
  X var byte
  C var byte
  me var byte
  me2 var byte
  lowertemp var byte
  raisetemp var byte
  k var byte
  i var byte
  deltat var byte
  this var byte
  buttons var byte
  tunnel var byte
  tunnel2 var byte
  tunnel1 var byte
  tunnel2add var byte
  tunnel1add var byte
  tunnel1up var byte
  tunnel2up var byte
  u var byte
  z var byte
  'push button setup (not interrupts)
  override  var PORTD.1  'set port for override button
  reset   var PORTD.0   'set port for reset button
'Port D and C setup, used for H-bridges and LEDs
ANSELD= %00000000   'turn off port d adc
TRISD= %00000011     'port D is all outputs for H-bridge and LEDs, except for 1
and 0 which are button inputs
ANSELB=%00000000    'turn off port b adc
TRISC=%00000000     'port c is all outputs for H-bridge and LEDs
'interrupt setup for portb.0
IOCBP=%00000001
intcon=%10010000
'set default temperatures for lower and raise temps
raisetemp= 20
lowertemp= 30
'variable initialization, have both tunnels in loop and all tunnels down
i=1
tunnel1=0
tunnel2=0
u=1
z=1
main:
    pause 1000 'allow time for LCD and other stuff to set up
    'run the AD conversion twice, first value is usually off for some reason
    for me=1 to 2
         adcin 5, adcVar  'read value from AN5
         'convert digital value to a temperature, code doesn't save fractions
         xx=adcVar*100/1023   'multiple digital value by 100 and divide by bits
         yy=adcVar*100//1023  'get the remainder from previous operation
         tt=yy*10/1023        'use the remainder to keep measurement resolution
         temp1=((xx*50+tt*5)-475)/10 'conversion to first two digits of temp
measurement
         temp2= ((xx*50+tt*5)-475)//10 'conversion to decimal value of temp
measurement
```

```
    next me
    'display temp measurements and raise/lower setpoints
    lcdout $fe, 1, "temp:", dec temp1, ".", dec temp2, "C"
    lcdout $fe, $C0, "raise:", dec raisetemp, "lower:", dec lowertemp

     'error loop if raise temp is higher than low temp, code will not work
    if (raisetemp > lowertemp) then
        lcdout $fe, 1, "error"
        goto main
    endif

    'wait to take next temp measurement
    'break up into short pauses so interrupt will appear sooner
    pause 1000
    pause 1000
    pause 1000
    pause 1000
    pause 1000

    'run AD conversion twice again to allow for appropriate value
    for me2=1 to 2
        adcin 5, adcVar     'same process for new temp measurement
        xxx=adcVar*100/1023
        yyy=adcVar*100//1023
        ttt=yyy*10/1023
        temp3=((xxx*50+ttt*5)-475)/10
        temp4= ((xxx*50+ttt*5)-475)//10
     next me2

     'compare measurements, not sure if will save negatives so ratio is used
     'won't save decimals so the ratio is multiplied by a 100,
     'deltat>100 means temperature increased, deltat<100 decrease
     deltat=(temp3*10+temp4)*100/(temp1*10+temp2)
     'show new temp and change in temp ratioon LCD
     lcdout $fe, 1, "Temp:", dec temp3, ".", dec temp4, "C"
     lcdout $fe, $C0, "delta t:", dec deltat
     pause 1000
     'if temp is increasing, above the raisetemp, and the loop is down (i=1)
     if (((temp3*10+temp4) > (temp1*10+temp2)) && (temp3 > raisetemp) && (i==1))
then
            high PORTD.2 'turn on led to show loop state
            i=0           ' loop is now up (i=0)
            'if tunnel1 is in the loop and tunnel 1 is down (u=1)
            if ((tunnel1 == 0) && (u==1)) then
                        high portC.0 'turn on LED to show motor 1 state
                        pause 100   'allow LED to turn on, needs this for some
reason
                        high portc.7 'turn buzzer/motor speed 1
                        pause 1000   'allow buzzer to sound before motor starts
                        'motor won't start cause its in brake mode
                        'set H-bride pins to turn motor one way
                        high portC.6 'H-bridge directional pin
                        low portC.4  'H-bridge directional pin
                        pause 3000   'allow motor to turn for 3 seconds
                        'set H-bridge pins back to brake mode
                        low portC.6
                        low portC.4
                        low portc.7 'turn buzzer/motor speed off
                        u=0         'motor 1 is now up
            endif

            'if tunnel2 is in the loop and tunnel 2 is down (z=1)
```

```
            'same procedure as above but with different pins for motor 2
            if ((tunnel2 == 0) && (z == 1)) then
                        high portC.1
                        pause 100
                        high portd.7
                        pause 1000
                        high portd.6
                        low portd.5
                        pause 3000
                        low portd.6
                        low portd.5
                        low portd.7
                        z=0
            endif
      endif 'this ends the "raising" automated loop segment
      'if temp is decreasing, below the lower temp, and the tunnel is up (i=0)
      if (((temp3*10+temp4) < (temp1*10+temp2)) && (temp3 < lowertemp ) && (i==0))
then
           low PORTD.2  'turn off loop LED
           i=1          'loop is now down
           'if tunnel 1 is in loop and tunnel 1 is up (u=0)
           if ((tunnel1 == 0) && (u==0)) then
                        'same procedure but reverse H-bridge directional pins
                       low portC.0
                       pause 100
                       high portc.7
                       pause 1000
                       low portC.6
                       high portC.4
                       pause 3000
                       low portC.6
                       low portC.4
                       low portc.7
                       u=1
          endif
          'if tunnel 2 is in loop and tunnel 2 is up (z=0)
          if ((tunnel2 == 0) && (z==0)) then
                         'same procedure but reverse H-bridge directional pins
                         low portC.1
                         pause 100
                         high portd.7
                         pause 1000
                         low portd.6
                         high portd.5
                         pause 3000
                         low portd.6
                         low portd.5
                         low portd.7
                         z=1

          endif


      endif 'this ends the "lowering" automated loop segment


      goto main 'loop back to the start

disable
myint:    'interrupt sequence
     'define internal interrupt variables at the start of each interrupt
```

```
    'this is important so that it doesn't jump to any of the keypad loops
    'note we were unable to call functions in the interrupt so the code is a bit
    'repititive
    tunnel=100
    tunnel1add=100
    tunnel2add=100
    tunnel1up=100
    tunnel2up=100
    this=1
while(this == 1) 'this is set up cause we had problems staying in the interrupt
          lcdout $Fe,1, "select override"
          lcdout $Fe,$C0, "select reset"
          buttons=100 'define variable to enter while loop
          while (buttons == 100) 'variable is only redefined if one of the buttons
is pressed
                if ((reset != 0) && (override != 1)) then
                     buttons=1 'leave loop, reset has been pressed
                endif
                if ((reset!= 1) && (override != 0)) then
                    buttons=2 'leave loop, override has been pressed
                endif
          wend


    if (buttons == 2) then  'if override has been pressed
          lcdout $Fe,1, "select tunnel"
          lcdout $Fe,$C0, "press tunnel #"
          tunnel=100 'definte variable to enter while loop
          while (tunnel == 100) 'variable only redefined if 1 or 2 is pressed
                  low col1 : high col2 : high col3
                  if (row1 == 0) then
                        tunnel=1 ' key 1 is pressed
                  endif
                  ' Check column 2
                  high col1 : low col2 : high col3
                  if (row1 == 0) then
                        tunnel=2     ' key 2 is pressed
                  endif
          wend


          lcdout $Fe,1, "tunnel:", dec tunnel    'print tunnel selected
          pause 1000
          if (tunnel == 1) then   'if tunnel 1 was selected, add or remove from
loop
                  lcdout $Fe,1, "add: 1"
                  lcdout $Fe,$C0, "remove: 2"
                  tunnel1add=100   'define variable to enter while loop
                  while (tunnel1add == 100) 'only leave loop when 1 or 2 is pressed
                        low col1 : high col2 : high col3
                        if (row1 == 0) then
                             tunnel1add=1 ' key 1 is pressed
                        endif

                        ' Check column 2
                         high col1 : low col2 : high col3
                        if (row1 == 0) then
                             tunnel1add=2     ' key 2 is pressed
                        endif

                  wend

                 if (tunnel1add=1) then
```

```
                        tunnel1=0 'add tunnel 1 into automated loop
                endif

                if (tunnel1add=2) then
                    tunnel1=1    'remove tunnel 1 from automated loop
                endif
                if (tunnel1add==1) then  'if tunnel is added
                    if ((u != i) && (i=1)) then  'if tunnel doesn't match loop and
loop is low, lower tunnel
                        low portC.0
                        pause 100
                        high portc.7
                        pause 1000
                        low portC.6
                        high portC.4
                        pause 3000
                        low portC.6
                        low portC.4
                        low portc.7
                        u=1    'tunnel is now low
                    endif
                    if ((u != i) && (i=0)) then 'if tunnel doesn't match loop and
loop is high, raise tunnel
                        high portC.0

                        pause 100
                        high portc.7
                        pause 1000
                        high portC.6
                        low portC.4
                        pause 3000
                        low portC.6
                        low portC.4
                        low portc.7
                        u=0   'tunnel is now high
                    endif

                endif

            endif

            if (tunnel == 2) then    'repeat above steps if tunnel 2 is pressed
                lcdout $Fe,1, "add: 1"
                lcdout $Fe,$C0, "remove: 2"
                tunnel2add=100
                'gosub readhex    'call subroutine
                while (tunnel2add == 100)
                    low col1 : high col2 : high col3
                    if (row1 == 0) then
                        tunnel2add=1 ' key 1 is pressed
                    endif
                    ' Check column 2
                    high col1 : low col2 : high col3
                    if (row1 == 0) then
                        tunnel2add=2     ' key 2 is pressed
                    endif
                wend

                if (tunnel2add=1) then
                    tunnel2=0
                endif
                if (tunnel2add=2) then
```

```
            tunnel2=1
        endif

        if (tunnel2add==1) then
            if ((z != i) && (i=1)) then
                low portC.1
                pause 100
                high portd.7
                pause 1000
                low portd.6
                high portd.5
                pause 3000
                low portd.6
                low portd.5
                low portd.7
                z=1
            endif
            if ((z != i) && (i=0)) then
                high portC.1
                pause 100
                high portd.7
                pause 1000
                high portd.6
                low portd.5
                pause 3000
                low portd.6
                low portd.5
                low portd.7
                z=0
            endif
        endif
endif


if ((tunnel1add == 2)) then  'if tunnel 1 is removed, raise or lower it
    lcdout $Fe,1, "up: 3"
    lcdout $Fe,$C0, "down: 6"
    tunnel1up=100  'define variable to enter while loop
    while (tunnel1up == 100) 'only leav
            high col1 : high col2 : low col3
            if (row1 == 0) then
                tunnel1up=1 ' key 3 is pressed
            endif
            if (row2 == 0) then
                 tunnel1up=2     ' key 6 is pressed
            endif

   wend
endif
'tunnel has been set to down
if ((tunnel1up == 2) && (u==0)) then  'lower tunnel 1 if tunnel is up
    low portC.0
    pause 100
    high portc.7
    pause 1000
    low portC.6
    high portC.4
    pause 3000
    low portC.6
    low portC.4
    low portc.7
    u=1
```

```
    endif
    'tunnel has been set to up
    if ((tunnel1up == 1) && (u==1))   then  'raise tunnel 1 if tunnel is down
        high portC.0
        pause 100
        high portc.7
        pause 1000
        high portC.6
        low portC.4
        pause 3000
        low portC.6
        low portC.4
        low portc.7
        u=0
    endif
    'repeat for tunnel 2
    if ((tunnel2add == 2)) then
        lcdout $Fe,1, "up: 3"
        lcdout $Fe,$C0, "down: 6"
        tunnel2up=100
        while (tunnel2up == 100)
            high col1 : high col2 : low col3
            if (row1 == 0) then
                tunnel2up=1 ' key 3 is pressed
            endif

            if (row2 == 0) then
                tunnel2up=2     ' key 6 is pressed
            endif
        wend
    endif

    if ((tunnel2up == 2) && (z==0)) then
        low portC.1
        pause 100
        high portd.7
        pause 1000
        low portd.6
        high portd.5
        pause 3000
        low portd.6
        low portd.5
        low portd.7
        z=1
    endif

    if ((tunnel2up == 1) && (z==1)) then
        high portC.1
        pause 100
        high portd.7
        pause 1000
        high portd.6
        low portd.5
        pause 3000
        low portd.6
        low portd.5
        low portd.7
        z=0
    endif
endif
```

```
if (buttons == 1) then  'if reset was pressed and not override
lcdout $FE,1,$FE,$C0,"Lower Temp:"
for T=1 to 2
        key=100 'variable to enter while loop, only changed by key press
        while (key == 100)
            low col1 : high col2 : high col3
            if (row1 == 0) then
                key=1 ' key 1 is pressed
            endif
            if (row2 == 0) then
                key=4 ' key 4 is pressed
            endif
            if (row3 == 0) then
                key=7    ' key 7 is pressed
            endif

            ' Check column 2
            high col1 : low col2 : high col3
            if (row1 == 0) then
                key=2     ' key 2 is pressed
            endif
            if (row2 == 0) then
                key=5       ' key 5 is pressed
            endif
            if (row3 == 0) then
                key=8       ' key 8 is pressed
            endif
            if (row4 == 0) then
                key=0         ' key 0 is pressed
            endif

            ' Check column 3
            high col1 : high col2 : low col3
            if (row1 == 0) then
                key=3      ' key 3 is pressed
            endif
            if (row2 == 0) then
                key=6      ' key 6 is pressed
            endif
            if (row3 == 0) then
                key=9      ' key 9 is pressed
            endif

       wend
       if (T == 1)  then 'first time through loop is tens digit
            numblink=key
            lcdout $FE, $14, dec numblink 'print digit
       endif
       if (T == 2) then 'second time is the ones digit
            numblinks=key
            lcdout $FE,$14,  dec numblinks 'print digit
       endif
next T

pause 1000
lcdout $fe, 1, "temp:", dec temp1, ".", dec temp2, "C"
lcdout $FE,$C0,"Raise Temp:"
for C=1 to 2
    key=100 'same procedure as before for raise temp
    while (key == 100)
       low col1 : high col2 : high col3
       if (row1 == 0) then
```

```
                key=1  ' key 1 is pressed
            endif

            if (row2 == 0) then
                key=4  ' key 4 is pressed
            endif
            if (row3 == 0) then
                key=7     ' key 7 is pressed
            endif

             'Check column 2
            high col1 : low col2 : high col3
            if (row1 == 0) then
                key=2      ' key 2 is pressed
            endif
            if (row2 == 0) then
                key=5        ' key 5 is pressed
            endif
            if (row3 == 0) then
                key=8        ' key 8 is pressed
            endif
            if (row4 == 0) then
                key=0         ' key 0 is pressed
            endif

           ' Check column 3
            high col1 : high col2 : low col3
            if (row1 == 0) then
                key=3       ' key 3 is pressed
            endif
            if (row2 == 0) then
                key=6       ' key 6 is pressed
            endif
            if (row3 == 0) then
                key=9       ' key 9 is pressed
            endif
        wend

        if (C == 1)  then
            numblink3=key
            lcdout $FE, $14, dec numblink3
        endif
        if (C == 2) then
            numblink4=key
            lcdout $FE,$14,  dec numblink4
        endif
    next C
    raisetemp=numblink3*10+numblink4
    lowertemp=numblink*10+numblinks
    endif
    intcon.1 = 0       'clear interrupt
    this=0

wend

resume
enable
```