**Rise & Shine**

***A Daylight-Sensing Automatic Coffee Maker***

****

**Will Carhart**

**Sean Fernandez**

Table of Contents

Background 3

Purpose 3

Project Description 4

Hardware Design 5

Software Design 8

Design Considerations 11

Gallery 12

Appendix 15

**Background**

Many Americans rely on a good cup o’ Joe to help them rise and shine in the morning. Reports by the Huffington Post and the Harvard T.H. Chan School of Public Health concluded that 54% of Americans drink at least one cup of coffee every day, with 65% of Americans drinking it with their breakfast. In addition, 58% of coffee drinkers brew their coffee the same exact way every single morning.

With a regular coffee maker, an American can brew his or her own cup of coffee at home in about 10 minutes. If the average American made a cup of coffee every single day from age 20 to age 85, he or she would spend 236,600 minutes waiting for coffee in his or her lifetime. That’s equivalent to 3943 hours, or 164 days, of waiting for coffee to be made. Granted, the average American can be busy with other interests while coffee is being made, but almost half a year of his or her life spent waiting of coffee is a long time.

What if there was a way for coffee to be made *as soon as* the average American was awake? What if instead of waiting 10 minutes every morning, he or she waiting 0 minutes?

**Purpose**

The purpose of this project is to design an embedded system to automate the coffee making process. We will design circuitry around the PIC18F4321, as well as complementary software, to accomplish this task.

**Project Description**

This project will operate in two separate parts: circuitry to control the coffee maker (the *Hardware*) and a program to automate the process (the *Software*). The project will utilize several design features to accomplish this task.

*Alarm Clock*

The project will feature an alarm clock. This clock will be a standard 12-hour digital clock, with the time format HH:MM:SS. The alarm clock will function per standard clock regulations, with one hour passing every sixty minutes, and one minute passing every sixty seconds. In addition, the clock will be able to determine AM vs. PM, and roll over accordingly when the clock changes from 11:59 to 12:00. The clock will be able to be set by the user.

*Human Interface*

The project will have a human interface in the form of a Liquid Crystal Display (LCD) and analog switches (buttons). The LCD will display the current time in the format HH:MM:SS. The time can be set by using the attached buttons. One button will be used as a debugging tool to pause program execution, and three other buttons will be used to set the clock’s time (one to increment the hours, one to increment the minutes, and one to increment the seconds).

*Coffee Maker*

The project will automate a Mr. Coffee Model CG13 Coffee Maker. The microcontroller will essentially bypass the analog switch on the coffee maker’s housing, and turn it on with a control signal.

*Daylight Sensor*

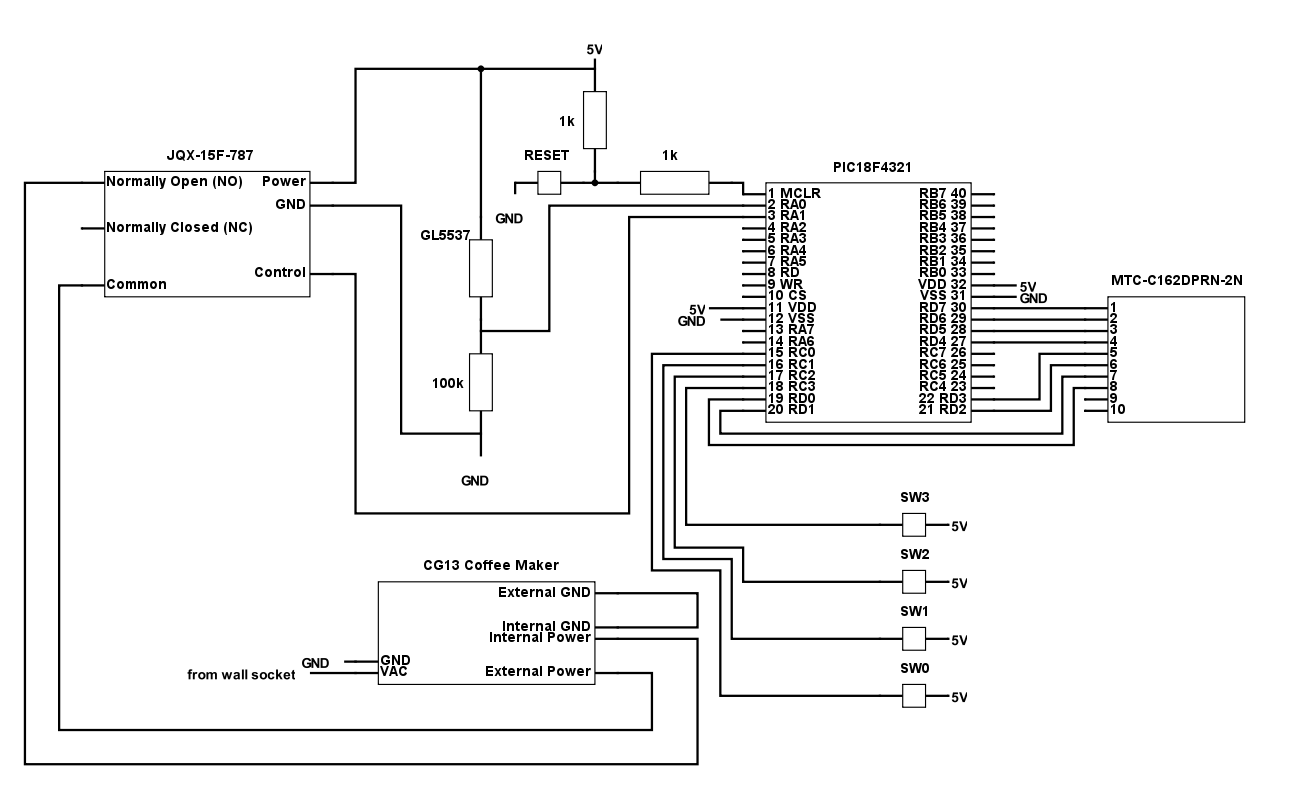
The project will use a daylight sensor to sense the brightness of the room. The daylight sensor will be left near a window or on a bedside table, and when it senses that the light is bright enough, the project will turn on the coffee maker. The hope is that the daylight sensor will sense when the day begins and start making coffee before the user wakes up, so that coffee is ready and prepared prior to the user entering the kitchen.

**Hardware Design**

**Hardware**

1. (1x) *Mr. Coffee* Model CG13 Coffee Maker
2. (1x) Shaanxi Qunli Electric Company Miniature Heavy Duty DC Electromagnetic Relay (JQX-15F-787)
3. (1x) GL5537 Photoresistor
4. (1x) Microchip PIC18F4321 Microcontroller
5. (1x) Microchip PICkit 3 In-Circuit Debugger (PG164130)
6. (1x) Kanda Kit – Main Board
7. (1x) Kanda Kit – Training Board
8. (1x) Kanda Kit - LCD Board
9. (1x) Truly LCD Module (MTC-C162DPRN-2N)
10. (2x) resistor
11. (1x) resistor

**Schematic**

*Figure 1: Rise & Shine schematic*

**Design**

We started by embedding our PIC18F4321 with a Kanda Kit, which included a main board and a training board. The main board has headers for Ports A-D, as well as a switch for Reset. The training board houses the switches and LEDs, as well as the hardware interface for the LCD Module.

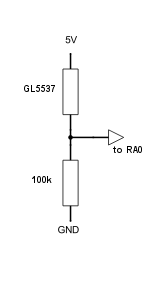
*Sensing*

The daylight sensor circuitry consists of the GL5537 photoresistor in series with a resistor. The GL5537 varies its impedance based on its environment’s brightness. The impedance value of the GL5537 are shown in Table 1.

*Table 1: GL5537 Illuminance and Resistance Values*

|  |  |  |
| --- | --- | --- |
| **Setting** | **Illuminance ()** | **Impedance ()** |
| Day | 10 | 5 |
| Night | 1 | 500 |

The brighter the light, the smaller the impedance of the GL5537. The daylight senor circuitry is shown in Fig. 2.

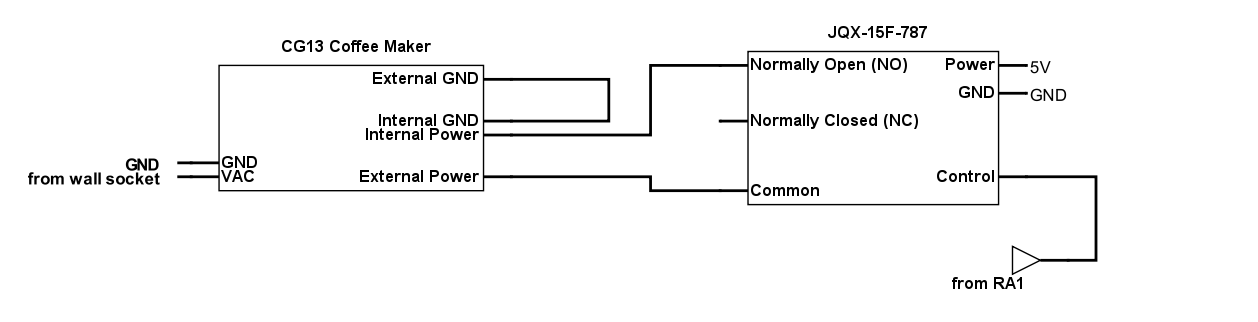


*Figure 2: Daylight sensor circuitry*

When the light is on (representing daytime), *RA0* would read a value of at least , as the impedance of the GL5537 would drop to , or 5% of the sensor’s total impedance. When the light is off (representing nighttime), *RA0* would read a value of almost (depending on the brightness of the room), as the impedance of the GL5537 would raise to at least , or 83.3% of the sensor’s total impedance.

*Controlling*

The CG13 Coffee Maker is controlled by a JQX-15F-787 Electromagnetic Relay. The CG13 is natively controlled by an analog Single Pole Single Throw (SPST) switch. Within the internal circuitry of the CG13, the SPST connects the *External Power* to *Internal Power* when thrown. The *Internal GND* and *External GND* pins are always shorted together. The controlling circuitry is shown in Fig. 3.



*Figure 3: JQX-15F-787 & CG13 control circuitry*

When it is time for the CG13 to turn on, the PIC18F4321 puts a *logic high* () on *RA1*, which causes the relay to connect the *Normally Open (NO)* and *Common* leads. This shorts the *External Power* and *Internal Power* pins on the CG13, causing it to turn on.

*Human Interface*

The “human-friendly” part of the hardware consists of the LCD Module and four analog push-button momentary switches. The LCD Module displays the current time in the format HH:MM:SS AM/PM. The user can change or set the time by using the momentary switches. The function of each switch is listed in Table 2.

*Table 2: Switch Functions & PIC18F4321 Locations*

|  |  |  |
| --- | --- | --- |
| **Switch** | **PIC18F4321 Pin** | **Function** |
| *SW0* | *RC0* | Pause program execution |
| *SW1* | *RC1* | Increment hours |
| *SW2* | *RC2* | Increment minutes |
| *SW3* | *RC3* | Increment seconds |

**Software Design**

**Implementation Details**

The software for the project has two primary functions:

1. Control alarm clock and timing
2. Sense input from daylight sensor circuitry and send corresponding output to electromagnetic relay

*Initializations*

Table 3 shows the initialization values for the software component of the program, as well as the function of the initializations.

*Table 3: Software initialization values & functions*

|  |  |  |
| --- | --- | --- |
| **Register/Bit Name** | **Value** | **Function** |
| \_XTAL\_FREQ | *8000000* | Defines the expected processor frequency (*8MHz*) for use in software, used for software delays |
| ADCON1 | *0x0C* | Sets pins *RA0* and *RA1* to analog |
| TRISA | *0x01* | Sets *RA0* input and *RA1* as output |
| TRISC | *0x00* | Sets PORTC as input (connected to switches) |
| TRISD | *0xFF* | Sets PORTD as output (connected to LCD Module) |
| IRCF | *0b111* | Sets PIC18F4321 processor to operate at *8 MHz* |
| CHS3:CHS0 | *0b0000* | Selects Channel0 for ADCON0 |
| ADFM | *0b1* | Right justifies ADC result |
| ACQT2:ACQT0 | *0b100* | Sets ADC acquisition time to *2TAD* |
| ADCS2:ADCS0 | *0b000* | Sets ADC conversion clock to |
| TMR0IE | *0b1* | Enables interrupt structure on TIMER0 |
| TMR0IF | *0b0* | Clears TIMER0 interrupt flag |
| GIE | *0b1* | Enables global interrupts |
| PSA | *0b1* | Neglects clock prescaler value for TIMER0 |
| T08BIT | *0b0* | Sets TIMER0 to *16bit* operation |
| T0CS | *0b0* | Enables internal clock for TIMER0 use |
| TMR0ON | *0b1* | Enables TIMER0 |

*Analog-Digital Conversion & Daylight Sensing*

The value of the daylight sensor (which will be in the range of ) will be read in on *RA0*. Given that the PIC18F4321 is a digital microprocessor, the analog value on *RA0* will be converted into a digital value using the PIC18F4321’s ADC process. Once the result is converted and scaled to the correct voltage range, it is used in the software. The software used the following activation function to control the CG13 based on the daylight sensor circuitry output:

*Alarm Clock Regulation*

The alarm clock is used in conjunction with the PIC18F4321’s TIMER0 timer. TIMER0 has a built-in interrupt structure, which occurs when the timer’s up-counter value rolls over to *0xFFFF*. Thus, we placed *0xB1C6* in the TIMER0 reset register. Given that each instruction cycle causes one tick in TIMER0, we have 20,0025 ticks in between TIMER0 interrupts. In addition, because the PIC18F4321’s internal clock was set to operate at *8MHz*, we can calculate the time in between TIMER0 interrupts in the following fashion:

Thus, a TIMER0 interrupt will be triggered every 0.1000125 seconds. We then used this to calculate the value of the current time, which was then manipulated into HH:MM:SS format.

**Function Descriptions**

char\* char2ASCII(unsigned char input)

The char2ASCII() function converts an unsigned char into an ASCII encoding for the given character. The parameter unsigned char input represents the unsigned char that will be converted into an ASCII encoding. The function returns a char\* that points to the result of the conversion.

void sendNibble(char nibble)

The sendNibble() function sends four bits to the LCD Module. The parameter char nibble represents the nibble (four bits) that will be sent to the LCD Module. The function returns void because it sends the nibble to the LCD Module and does not need to return anything. The implementation of this function adheres to the following steps:

1. Shift each of the four bits onto *RC7, RC6, RC5,* and *RC4* to be transferred to the LCD Module.
2. Pulse *RC2* in a low-high-low fashion for at least to pulse the LCD *E-Clock*, which sends the nibble to the LCD Module.

void sendByte(char data)

The sendByte() function send eight bits to the LCD Module. The parameter char data represents the byte (eight bits) that will be sent to the LCD Module. The function returns void because it sends the byte to the LCD Module and does not need to return anything. The implementation of this function adheres to the following steps:

1. Send top nibble of byte to LCD by calling sendByte().
2. Send bottom nibble of byte to LCD by calling sendByte().

void lcdAdd(char text[])

The lcdAdd() function adds a character or string to the LCD screen, adding it subsequent to the current cursor location. The parameter char text[] represents the string that will be added to the LCD screen. The function returns void because it adds text to the LCD screen and does not need to return anything. The implementation of this function adheres to the following steps:

1. Set the LCD Module to data mode.
2. Call sendByte() to print the string to the LCD screen one character at a time.

void lcdWrite(char text[])

The lcdWrite() function adds a character or string to the LCD screen, after resetting the display and cursor. The parameter char text[] represents the string that will be added to the LCD screen. The function returns void because it adds text to the LCD screen and does not need to return anything. The implementation of this function adheres to the following steps:

1. Set the LCD Module to program mode.
2. Clear the LCD screen
3. Reset the LCD cursor.
4. Set the LCD Module to data mode.
5. Call sendByte() to print the string to the LCD screen one character at a time.

void interrupt timerReset(void)

The timerReset() function serves as the TIMER0 interrupt service routine for the software. The function returns an interrupt that occurs when the TIMER0 buffer overflows to *0xFFFF*. The implementation of this function adheres to the following steps:

1. Move *0xB1* into TMR0H.
2. Move *0xC6* into TMR0L.
3. Clear the TIMER0 interrupt flag.
4. Increment time.
5. Recalculate values for hours, minutes, seconds, decimals, colons, and AM/PM select.

**Design Considerations**

**Hardware Considerations**

The first challenge we faced was designing the daylight sensor circuitry. We originally set out with a much more complex model that involved BJTs and other passive elements, but after revising our design multiple times, we determined that the best solution was the simplest one. We designed our daylight circuitry with our GL5537 photoresistor and a single resistor. The circuit works as a voltage divider between the GL5537 and resistor, which allowed us to pull accurate, stable data from the sensor at any time.

Our next challenge was building an interface with the CG13 Coffee Maker. Upon disassembling the internal circuitry of the model, we discovered that most of the hardware could be left untouched, and that we only needed to modify the switch. The switch that came integrated with the CG13 was a SPST switch, which meant we simply needed to bypass it. Our first attempt was to simply apply the voltage different between the *External GND* and *External Power* to the *Internal GND* and *Internal Power* pins of the switch. However, we discovered this would not work, as the voltage difference was not the problem, it was the current. To activate the heating element of the CG13, the integrated switch effectively passed of current to the device. Rather than figuring out a way to apply this current from the underpowered PIC18F4321, we used the JQX-15F-787 Electromagnetic Relay to short *External Power* and *Internal Power* when we needed to activate the CG13.

**Software Considerations**

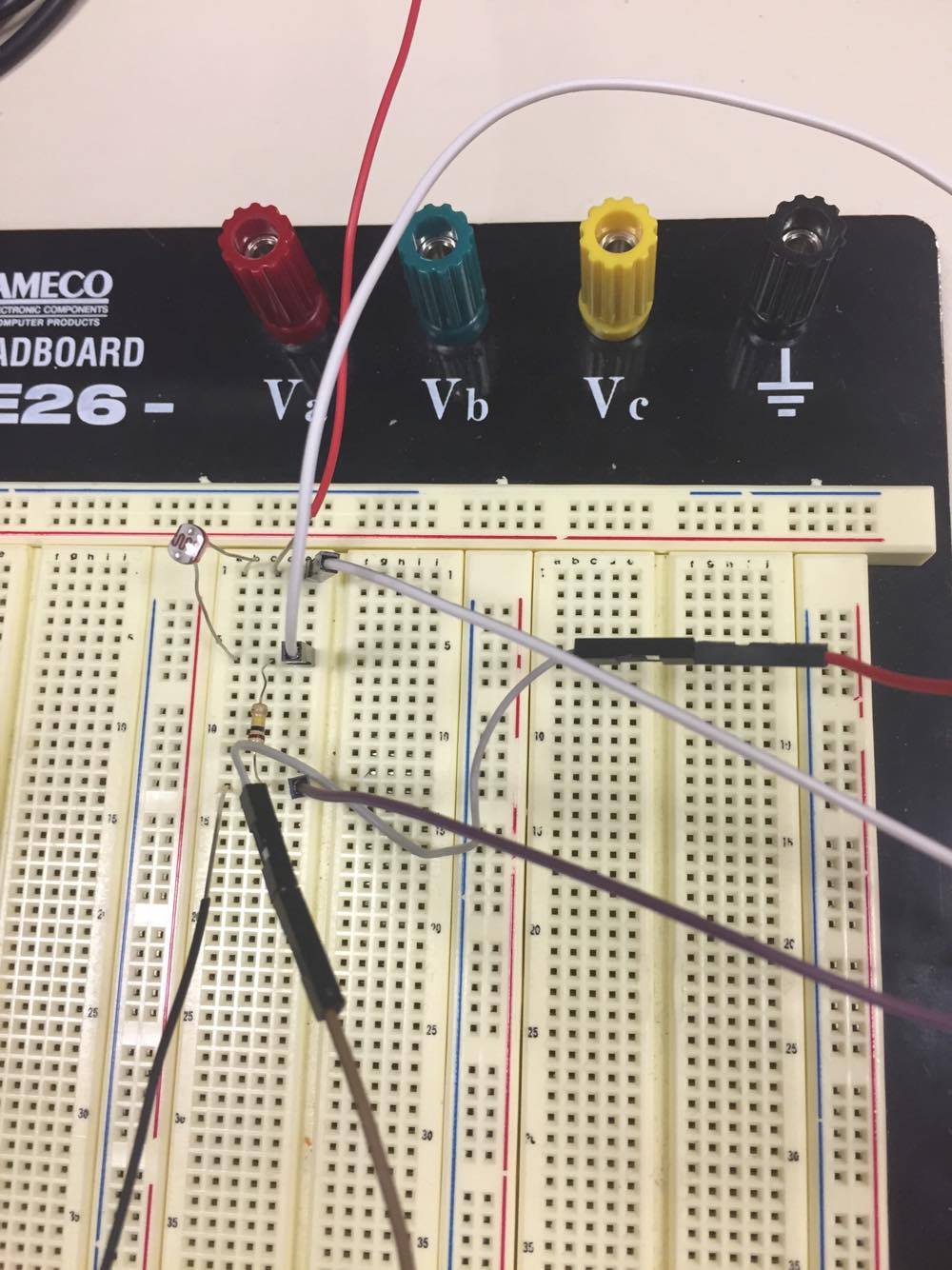
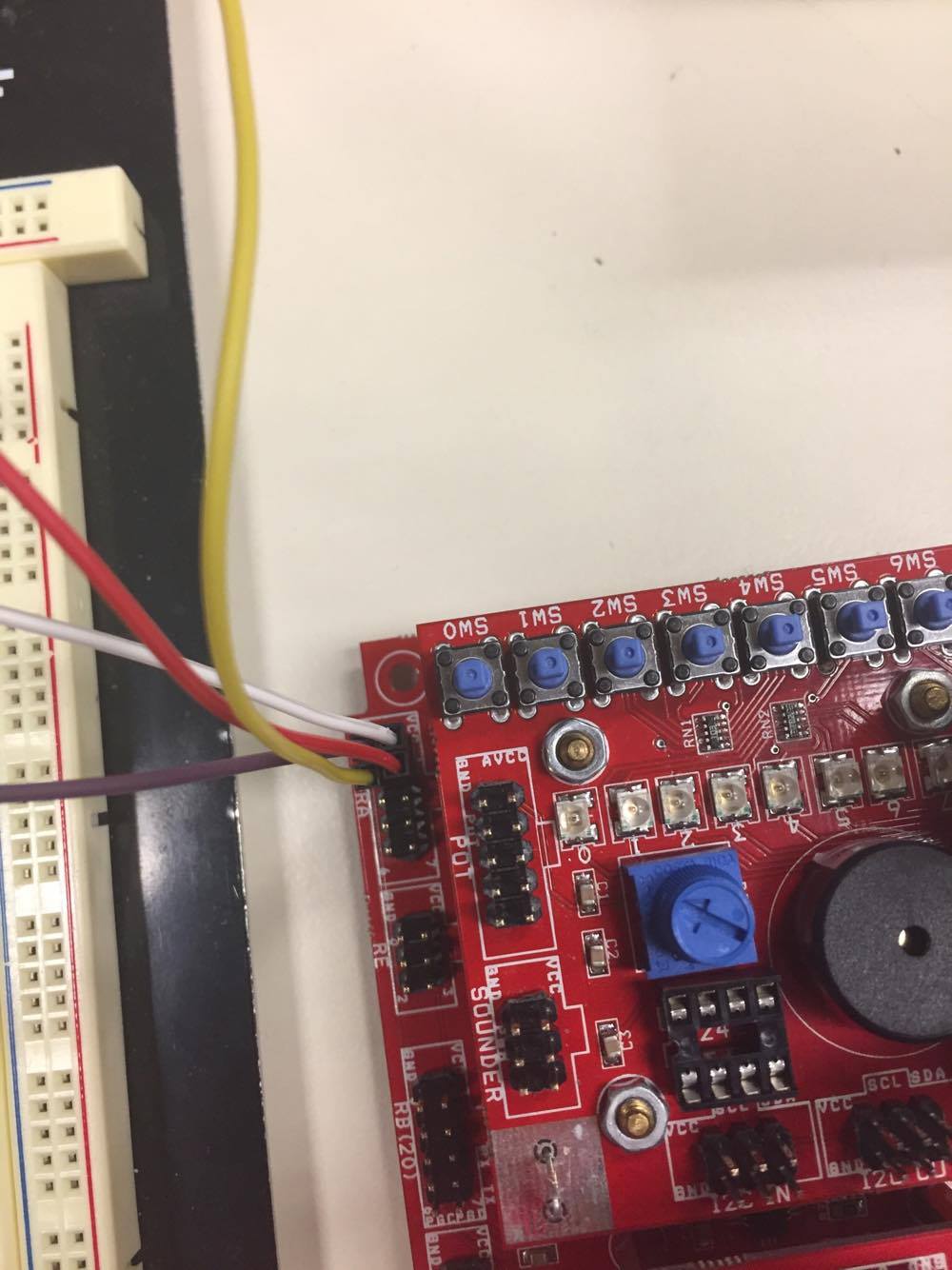
Our first challenge was determining how to read in the value from the daylight sensor circuitry. We hooked the sensor up to *RA0*, realizing that we needed to convert the analog value from the pin into a digital value that we could manipulate with our software. This led us to create the ADC solution that we did.

Another challenge was creating an accurate clock for our human interface. We had to make sure the OSCCON value for frequency matched the \_XTAL\_FREQ definition, so that hardware and software agreed on the oscillator frequency. Once that was set, we simply computed the correct values (see *Software Design*) that needed to be moved into the TIMER0 interrupt registers.

**Gallery**

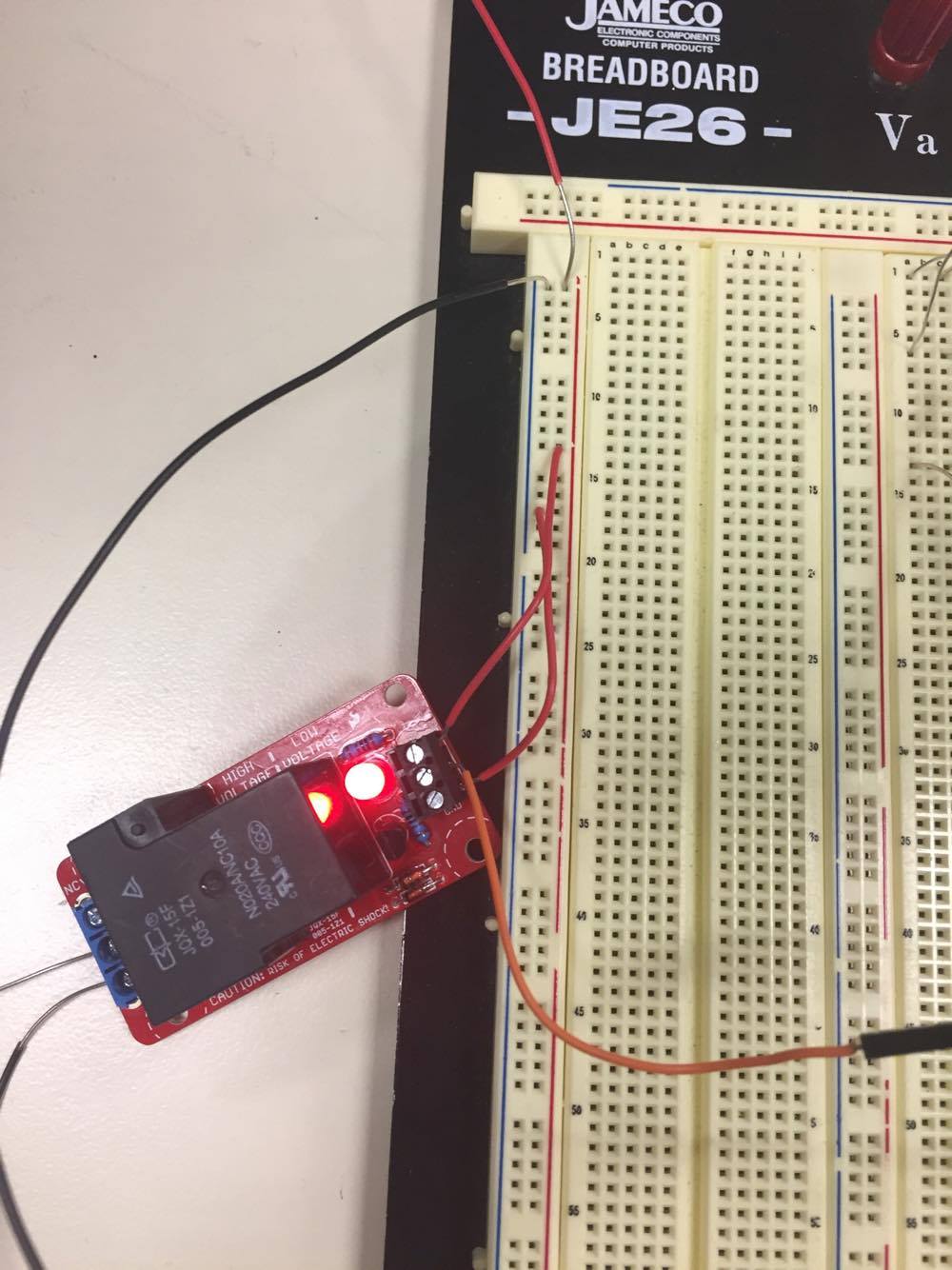


*Figure 4: PIC18F4321 on the Kanda Board, showing LCD with clock output*



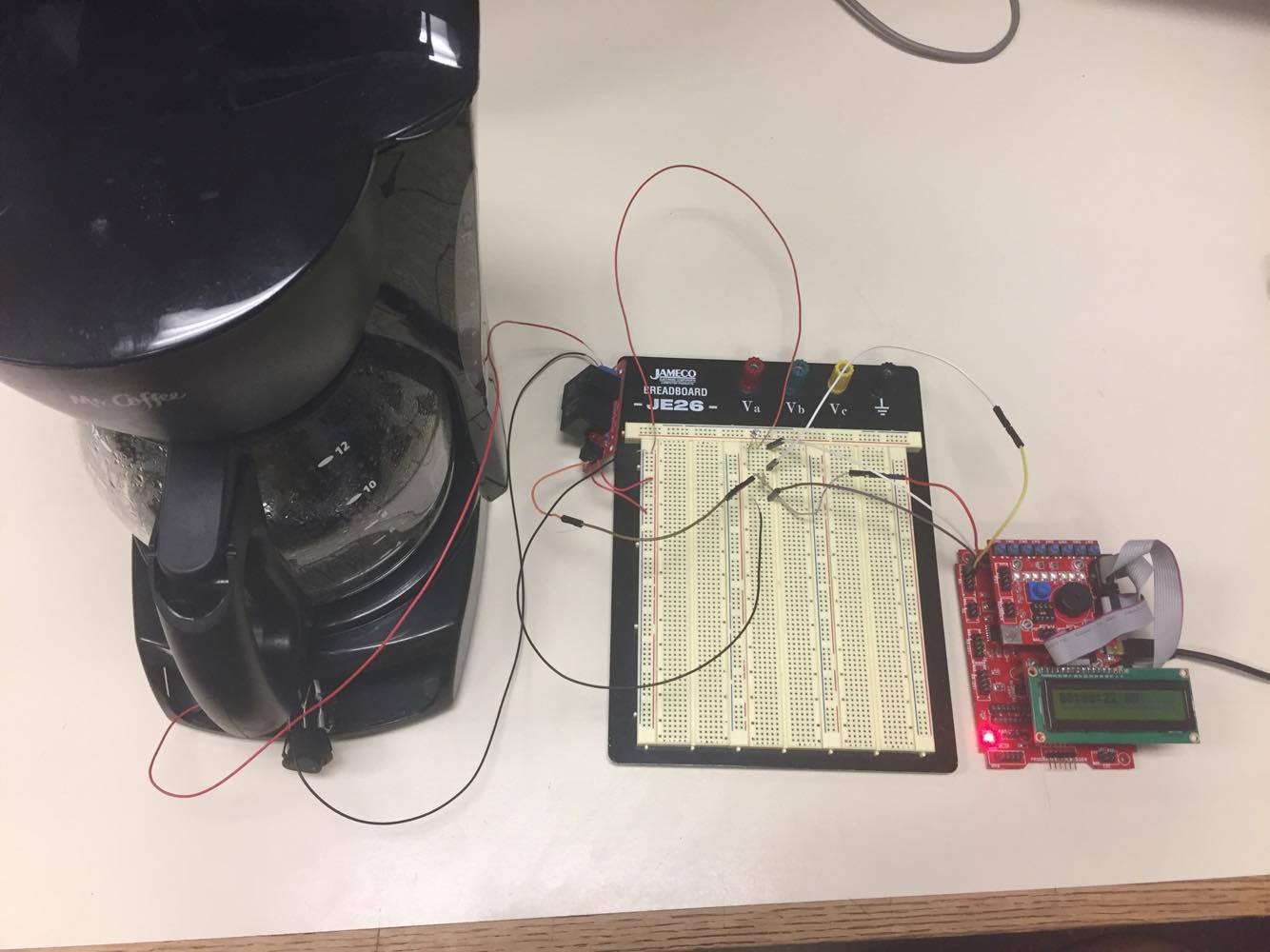
*Figure 6: RA0 used as input and RA1 used as output*

*Figure 5: Daylight sensor circuitry*

****

*Figure 7: Bypass of original switch on EG13 Coffee Maker*

*Figure 8:* JQX-15F-787 e*lectromagnetic relay circuitry*

****

*Figure 9: Full view of Rise & Shine*

**Appendix**

**Source Code**

/\*

\* ELEC 310 Final Project

\* Authors: Will Carhart, Sean Fernandez

\* Date: May 15th, 2017

\*/

// include necessary libraries

#include <xc.h>

#include <p18f4321.h>

#include <stdio.h>

// configure PIC18F4321

#pragma config OSC = INTIO2

#pragma config WDT = OFF

#pragma config LVP = OFF

#pragma config BOR = OFF

// internal frequency definition

#define \_XTAL\_FREQ 8000000

// define local aliases

#define C0 LATCbits.LC0

#define C1 LATCbits.LC1

#define C2 LATCbits.LC2

#define C3 LATCbits.LC3

#define C4 LATCbits.LC4

#define C5 LATCbits.LC5

#define C6 LATCbits.LC6

#define C7 LATCbits.LC7

// forward-declare function prototypes

**void** sendNibble(**char** nibble);

**void** sendByte(**char** data);

**void** lcdWrite(**char** text[]);

**char**\* char2ASCII(unsigned **char** temp);

**void** lcdAdd(**char** text[]);

// define globals

unsigned **long** time = 0x00000000;

unsigned **char** hours = 0x00;

unsigned **char** minutes = 0x00;

unsigned **char** seconds = 0x00;

unsigned **char** decimals = 0x00;

**int** colons = 1;

**int** AMPM = 1;

**void** main(**void**){

// set up analog/digital, input/output

ADCON1 = 0x0C;

TRISA = 0x01;

TRISC = 0x00;

TRISD = 0xFF;

// set up processor frequency

OSCCONbits.IRCF = 0b111;

// select channel on ADCON0

ADCON0bits.CHS0 = 0;

ADCON0bits.CHS1 = 0;

ADCON0bits.CHS2 = 0;

ADCON0bits.CHS3 = 0;

// right justify ADC result

ADCON2bits.ADFM = 1;

// acquisition time = 2TAD

ADCON2bits.ACQT0 = 1;

ADCON2bits.ACQT1 = 0;

ADCON2bits.ACQT2 = 0;

// set clock to Fosc/2

ADCON2bits.ADCS0 = 0;

ADCON2bits.ACQS1 = 0;

ADCON2bits.ACQS2 = 0;

// interrupt setup

INTCONbits.TMR0IE = 1; // enable interrupt on TIMER0

INTCONbits.TMR0IF = 0; // clear TIMER0 interrupt flag

INTCONbits.GIE = 1; // enable global interrupts

// TIMER0 setup

T0CONbits.PSA = 1; // do not assign prescaler to TIMER0

T0CONbits.T08BIT = 0; // use 16bit operation mode for TIMER0

T0CONbits.T0CS = 0; // use internal clock for TIMER0

T0CONbits.TMR0ON = 1; // turn on TIMER0

/\* LCD setup \*/

\_\_delay\_ms(15);

// set LCD to program mode

PORTCbits.RC0 = 0;

PORTCbits.RC1 = 0;

sendNibble(0b0011);

\_\_delay\_us(4100);

sendNibble(0b0011);

\_\_delay\_us(100);

sendNibble(0b0011);

\_\_delay\_us(5);

// set LCD to 4bit operation mode

sendNibble(0b0010);

\_\_delay\_us(42);

// configure LCD to 1 line display, 5x10 dots

sendByte(0b00100100);

\_\_delay\_us(42);

// turn on display, disable cursor

sendByte(0b00001100);

\_\_delay\_us(42);

// clear display

sendByte(0b00000110);

\_\_delay\_us(42);

// start time at 1:00 AM

time = 4320000;

// hardware initializations

ADCON0bits.ADON = 1;

PORTAbits.RA1 = 0;

// main program loop

**float** result, volt;

**while** (1) {

// A-->D conversion

ADCON0bits.GO = 1;

**while** (GO == 1);

result = ADRESL;

volt = (result / 255) \* 5;

// turn on coffe maker if bright enough

**if** (volt >= 3.0) {

PORTAbits.RA1 = 1;

} **else** {

PORTAbits.RA1 = 0;

}

// use buttons for setting time

**if** (!PORTDbits.RD1) {

time += 360000;

\_\_delay\_ms(50);

}

**if** (!PORTDbits.RD2) {

time += 6000;

\_\_delay\_ms(50);

}

**if** (!PORTDbits.RD3) {

time += 100;

\_\_delay\_ms(50);

}

// output time onto LCD

\_\_delay\_ms(10);

lcdWrite(char2ASCII(hours));

**if** (colons) {

lcdAdd(":");

} **else** {

lcdAdd(" ");

}

lcdAdd(char2ASCII(minutes));

**if** (colons) {

lcdAdd(":");

} **else** {

lcdAdd(" ");

}

lcdAdd(char2ASCII(seconds));

lcdAdd(" ");

**if** (AMPM) {

lcdAdd("A");

} **else** {

lcdAdd("P");

}

lcdAdd("M");

// used for pausing execution

**if** (!PORTDbits.RD0) {

INTCONbits.TMR0IE = 0;

\_\_delay\_ms(5);

**while**(!PORTDbits.RD0);

\_\_delay\_ms(5);

**while**(PORTDbits.RD0);

\_\_delay\_ms(5);

**while**(!PORTDbits.RD0);

INTCONbits.TMR0IE = 1;

}

}

}

**char**\* char2ASCII(unsigned **char** input) {

**char** disp[3];

disp[0] = input / 10;

disp[1] = input % 10;

disp[0] += '0';

disp[1] += '0';

disp[2] = '\0';

**return** disp;

}

**void** sendNibble(**char** nibble) {

PORTCbits.RC7 = (nibble >> 3) & 1;

PORTCbits.RC6 = (nibble >> 2) & 1;

PORTCbits.RC5 = (nibble >> 1) & 1;

PORTCbits.RC4 = nibble & 1;

// pulse E-Clock

PORTCbits.RC2 = 1;

\_\_delay\_us(1);

PORTCbits.RC2 = 0;

}

**void** sendByte(**char** data) {

sendNibble(data >> 4);

sendNibble(data & 0xF);

}

**void** lcdAdd(**char** text[]) {

// set LCD to data mode

PORTCbits.RC0 = 1;

// print string one char at a time

**for** (**int** i = 0; text[i] != 0; i++) {

sendByte(text[i]);

\_\_delay\_us(46);

}

}

**void** lcdWrite(**char** text[]) {

// set LCD to program mode

PORTCbits.RC0 = 0;

// clear display

sendByte(0b00000001);

\_\_delay\_us(1640);

// reset cursor

sendByte(0b00000010);

\_\_delay\_us(1640);

// set LCD to data mode

PORTCbits.RC0 = 1;

// print string one char at a time

**for** (**int** i = 0; text[i] != 0; i++) {

sendByte(text[i]);

\_\_delay\_us(46);

}

}

/\*

\* interrupt routine -- update time

\*/

**void** interrupt timerReset(**void**) {

asm("MOVLW 0xB1");

asm("MOVWF TMR0H");

asm("MOVLW 0xC6");

asm("MOVWF TMR0L");

asm("BCF INTCON, 2"); //clear interrupt flag

++time;

hours = ((time / 360000) % 12) + 1;

minutes = (time % 360000) / 6000;

seconds = ((time % 360000) % 6000) / 100;

decimals = (((time % 360000) % 6000) % 100);

**if** (time % 50 == 0) {

colons = (colons + 1) % 2;

}

**if** (time % 72000 == 0) {

AMPM = (AMPM + 1) % 2;

}

}

**Data Sheets**

The following documents are data sheets for the hardware used for this project. They are listed in the following order.

1. Microchip PIC18F4321 (omitted for brevity, can be accessed at http://ww1.microchip.com/downloads/en/DeviceDoc/39689f.pdf)
2. Truly LCD Module MTC-C162DPRN-2N
3. GL5537 Photoresistor
4. Shaanxi Qunli Electric Company Miniature Heavy Duty DC Electromagnetic Relay JQX-15F-787