PLANT ROBOT DOCUMENTATION

EE2361 Final Project

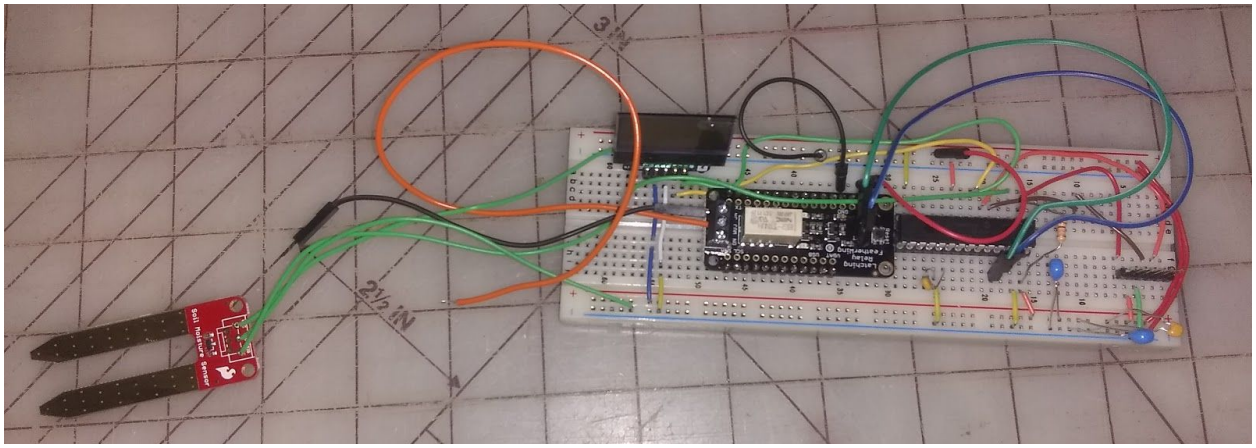
Amethyst O'Connell
Roman Woolery
Junchi Feng
Scott Deyo

Introduction

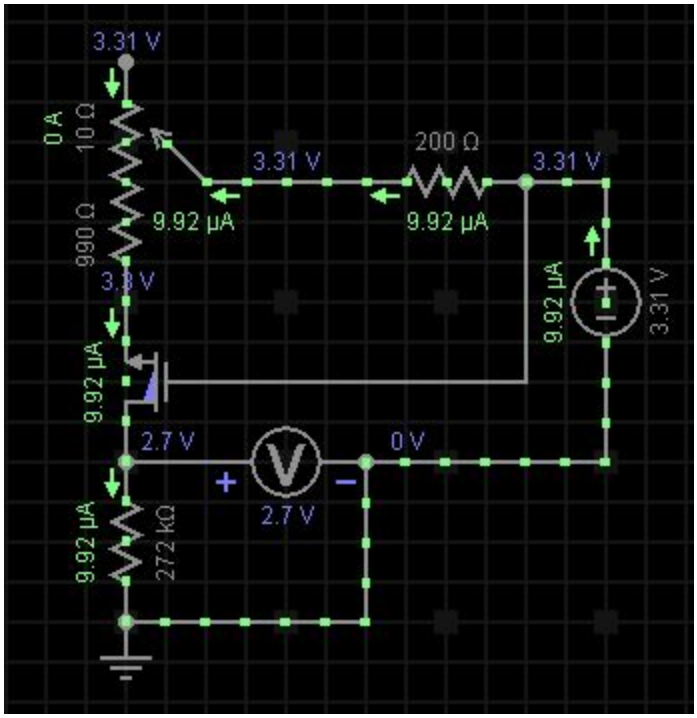
The Plant Robot uses the PIC24FJ64GA002 microcontroller to measure the moisture in the soil of a plant, and to water it accordingly. The code utilizes a single library roughly divided into three sections, display, flow control, and buffer. The display portion of the library runs the LCD screen which we use to display the percent moisture. The flow control portion of the library controls the relay that opens the solenoid. The buffer portion of the library records previous values of the sensor.

Hardware description - what device(s), part numbers, links, etc.

Sparkfun Soil Sensor, https://www.sparkfun.com/products/13322
Sitronix ST7032 LCD,
http://www.sitronix.com.tw/sitronix/product.nsf/Doc/ST7032?OpenDocument
Adafruit Latching 3V Relay, https://www.adafruit.com/product/2923
Adafruit Solenoid Flow Valve, https://www.adafruit.com/product/997
Power supply for relay
Large green watering can



The breadboard can be hooked up something like this. Alternatively, Altium or similar can be used to produce a printed circuit board in order to handle this task.



This is a rough circuit diagram of how the Sparkfun soil sensor works. It is a resistive moisture sensor, so, voltage passes between the two prongs of the sensor via electrolytes in the water/ soil -- the more conductive electrolytes, the higher the voltage output.

Full documentation - All public functions (arguments and outputs)

The Display portion of the library is an adaptation of the ADC/LCD combination of Lab 6, it reads and writes to the LCD screen using several functions.

```
// Display Functions
void delay(unsigned long int x){
// Function: delay
//  ---------------------------
// Delays the function for a number of milliseconds (x) using
assembly code. Used in lcd_init.
    unsigned long int y; //x is # of ms
    for(y=0; y<x; y++){
        asm("repeat #15998");
        asm("nop");
    }
}

void lcd_cmd(char command) {
// Function: lcd_cmd
//  ---------------------------
// Sends a command to the LCD screen
    I2C2CONbits.SEN = 1;
    while(I2C2CONbits.SEN == 1);
```

```
    IFS3bits.MI2C2IF = 0;
    I2C2TRN = 0b1111100;

    while(IFS3bits.MI2C2IF == 0);
    IFS3bits.MI2C2IF = 0;
    I2C2TRN = 0;

    while(IFS3bits.MI2C2IF == 0);
    IFS3bits.MI2C2IF = 0;
    I2C2TRN = command;

    while(IFS3bits.MI2C2IF == 0);
    IFS3bits.MI2C2IF = 0;
    I2C2CONbits.PEN = 1;
    while(I2C2CONbits.PEN == 1);
    IFS3bits.MI2C2IF = 0;
}

void lcd_init(void){
// Function: lcd_init
//  ---------------------------
// Initializes the LCD screen to be used in displaying information
    delay(50);
    lcd_cmd(0b00111000); // function set, normal instruction mode
    lcd_cmd(0b00111001); // function set, extend0000); // contrast
C3-C0
    lcd_cmd(0b01011110); // Ion, Bon, C5-C4ed instruction mode
    lcd_cmd(0b00010100); // interval osc
    lcd_cmd(0b01110000); // contrast C3-C0
    lcd_cmd(0b01011110); // Ion, Bon, C5-C4
    lcd_cmd(0b01101100); // follower control
    delay(200);
    lcd_cmd(0b00111000); // function set, normal instruction mode
    lcd_cmd(0b00001100); // Display On
    lcd_cmd(0b00000001); // Clear Display
    delay(2);
}

void lcd_setCursor(char x, char y){
// Function: lcd_setCursor
//  ---------------------------
// Sets the position for the cursor to write new letters based on
the numbers in x and y.
    int location = 0x40*y+x; //0x40 * row + column
    int coords = location + 0x80;
    lcd_cmd(coords);
}
```

```
void lcd_printChar(char myChar){
// Function: lcd_printChar
//  -------------------------
// Prints a character (myChar) onto the LCD screen

    I2C2CONbits.SEN = 1;    //Initiate Start condition
    while(I2C2CONbits.SEN == 1);  // SEN will clear when Start Bit
is complete
    IFS3bits.MI2C2IF = 0;
    I2C2TRN = 0b01111100; // 8-bits consisting of the slave address
and the R/nW bit

    while(IFS3bits.MI2C2IF == 0);
    IFS3bits.MI2C2IF = 0;
    I2C2TRN = 0b01000000; // 8-bits consisting of control byte /w
RS=1

    while(IFS3bits.MI2C2IF == 0);
    IFS3bits.MI2C2IF = 0;
    I2C2TRN = myChar; // 8-bits consisting of the data byte

    while(IFS3bits.MI2C2IF == 0);
    IFS3bits.MI2C2IF = 0;
    I2C2CONbits.PEN = 1;
    while(I2C2CONbits.PEN == 1); // PEN will clear when Stop bit is
complete
}

void lcd_printStr(const char *s){
// Function: lcd_printStr
//  -------------------------
// Prints a string onto the LCD screen. Takes in a string pointer.
    int i;
    for(i=0; i<strlen(s); i++){
        if(i>7){
            break;
        }
        lcd_printChar(s[i]);
    }
}
```

The FlowControl portion of the library controls the flow of the water to the plant, by using firmware to control the necessary water flow-related hardware.

In the flowControlInit function, using bitmasks, RB14 and RB15 are ensured as outputs, with RB15 used to set a 3V relay ON, and with RB14 used to turn it OFF. It also ensures that both outputs start low.

The waterTime function takes an unsigned int as its argument, water_time_in_seconds, which is the watering time in seconds. Since our water flow is controlled by a 12V solenoid that opens and closes the water valve, and we have a 3.3V power supply, we need a way to connect and disconnect 12V to the solenoid. We do this with a 3V relay -- a 3V-controlled electromechanical switch. When the function is called, a 10 ms pulse is sent to the relay's SET control. The relay is a latching model, which means it only needs a 10ms pulse on its SET pin to turn it ON, and a 10ms pulse on its UNSET pin to turn it OFF. The function pauses for the argued number of seconds, and then sends a 10ms pulse to UNSET. The advantage of this is low power -- a power MOSFET is not held on, nor the output pins of the micro. The disadvantage is that it uses two pins instead of one. When SET/ ON, 12V is connected to the solenoid, which pushes the water valve open, watering the plant. When UNSET/ OFF, power is disconnected from the solenoid, which retracts and lets the valve shut, turning off the water.

```
// Flow Control Functions
void flowControlInit(void)
// Function: flowControlInit
//  ---------------------------
// Initializes the pins used in flow control to work the relay for
the solenoid
{
    AD1PCFG &= 0x9fff; // all digital
    TRISB &= 0x3FFF;     // make sure RB0 and RB1 are outputs
    LATB &= 0x3FFF;         // make sure they start low
}

void pause(uint8_t pause_in_seconds)
// Function: pause
//  ---------------------------
// Delays the function for a number of seconds (pause_in_seconds).
Used in the waterTime function.
{
    int i = 0;
    while(i < pause_in_seconds)
    {
        delay(1000); // there are 1000 milliseconds in a second
    }
}

void waterTime(uint8_t water_time_in_seconds)
// Function: waterTime
//  ---------------------------
// Opens the solenoid valve to water the plant for a number of
```

```
seconds (water_time_in_seconds).
{
    uint32_t i;

    // Latch 'SET' on relay to turn flow solenoid ON
    LATB = 0x8000;          // RB0 latches solenoid ON
    for(i = 0; i < 160000; i++)asm("nop");   // keep high >10ms to
latch
    LATB = 0;          // Turn RB0 OFF to conserve power; solenoid
remains latched ON

    pause(water_time_in_seconds);   // Water keeps flowing

    // Latch 'UNSET' on relay to turn flow solenoid OFF
    LATB = 0x4000;          // RB1 latches solenoid OFF
    for(i = 0; i < 160000; i++)asm("nop");
    LATB = 0;          // ... RB1 is turned off to conserve power;
solenoid remains latched OFF
}
```

Basic usage example - bare minimum to test the functionality of the hardware

Holding onto the Sparkfun Soil Sensor so it detects different levels of hand moisture. The value of this moisture which is converted to a percentage will then show up on the LCD, and if below a certain number trigger the valve to open, such as 0% when the sensor isn't touching anything.

Advanced usage example - covering all the functions and features

In the Main function, this voltage reading is converted to percent moisture using a calibration value obtained by dipping the sensor in water and reading the voltage. Using the display portion of the library, this voltage is made into a string, and then passed to the LCD with a short message: "Too Dry", "Bit Wet", etc.

Putting the Sparkfun Soil Sensor in soil of a plant, with the watering time, moisture thresholds, and delay between moisture checks adjusted to work for a specific plants needs. The Sensor will send voltage corresponding to an average moisture value from a buffer to the PIC24 that will then be converted to a percentage and displayed on the Sitronix ST7032 LCD via ADC using the same library as Lab 6 with a different conversion calibrated for the sensor. Qualitative comments will also be displayed such as too wet, bit wet, normal, bit dry, too dry from the main loop. If too dry (below 20%), the valve will be told to move to allow water to drip on the plant for a set amount of time, specified with an integer representing seconds in the waterTime() function in the flow control library. This valve is moved using a connection to the Latching 3V Relay. This valve will be placed above the plant, so controlled watering will continue to occur until soil has reached a certain level of moisture, and continuously check if watering needs to be done again.

```
unsigned long int adValue; //(Analog to Digital) Value from the ADC
buffer
char adStr[50]; // (Analog to Digital) String for use with sprintf
below

void __attribute__((interrupt, auto_psv)) _ADC1Interrupt(){
    IFS0bits.AD1IF = 0; //read voltage from A0 for 16 times a second
    adValue = ADC1BUF0;
}

int main(void){
    setup();
    float curVal;

    while(1){
        lcd_setCursor(0,0);
        curVal = (3.3/1024)*adValue*100/2.6; //2.6 is a calibration
value, 2.6 volts, the maximum voltage

        sprintf(adStr, "%6.2f %%", curVal);
        lcd_printStr(adStr);

        lcd_setCursor(0,1);
        if(curVal >= 80)      lcd_printStr("Too wet"); // Percent
values in terms of watering
        else if(curVal >= 60) lcd_printStr("Bit wet"); // These
thresholds could be changed
        else if(curVal >= 40) lcd_printStr("Normal "); // For
different types of plants
        else if(curVal >= 20) lcd_printStr("Bit dry");
        else if(curVal >= 0){
            lcd_printStr("Too dry");
            waterTime(3);
        }
        putVal(getAvg());
        delay(100);
    }

    return 0;
}
```