A Simple Automated Home Garden Irrigation System Using PIC16F877A

Yusuf Qwareeq and Osama Abuhamdan
Computer Engineering Department, School of Engineering
The University of Jordan
yusufqwareeq@outlook.com, osamaabuhamdan@yahoo.com

I. Introduction

This project represents an environment-aware fully-automated home garden irrigation system using PIC16F877A that is supposed to manage a garden with two zones, each containing a different plant. The system has five sensors (potentiometers), two pumps (LED's) and an LCD. By default, the system will decide what to do based on the sensors' readings for each zone. The user is also able to control the system using four push-buttons. One of which is to switch between automatic and manual mode. Manual mode allows the user to turn the two pumps on or off using two other push-buttons regardless of the sensors' readings. An additional button is placed to allow the user to choose which zone will have its data displayed on the LCD. A red, 'frost alert' LED flashes when the temperature drops below 5 °C.

II. System Description

Upon start-up, the system defaults to automatic mode where pumps 1 and 2 are controlled by the system based on two rule-based algorithms (kindly refer to Table 1 and Table 2 in the project handout for more information) and 'AUTO MODE' will be displayed on the first line of the LCD. When the user switches to manual mode by pressing the 'Change Mode' push-button, they will be able to control the two pumps using two push-buttons 'Pump 1' and 'Pump 2', effectively overriding the auto-irrigation algorithm (pressing these two push-buttons does nothing while in automatic mode). Additionally, 'MANUAL MODE' will be displayed on the first line of the LCD. Pressing the 'Change Mode' push-button again will switch back to automatic mode.

Regardless of the set mode, the system will continuously process the five sensors' readings and display them on the second line of the LCD. Air temperature will be displayed first, soil humidity second and finally soil pH level. By default, the system displays the readings of zone 1. Pressing the 'Change Zone' push-button will display the readings of zone 2 instead. Pressing it again reverts back to displaying zone 1's readings.

Actual sensors were not used due to their high cost and difficulty in verifying the validity of the system's outputs. Instead, potentiometers were used as inputs to the A/D channels. A change in the environment is simulated by turning the knobs of said potentiometers. Each sensor has its own transfer function that was realized into code. Please refer to Figures 2, 3 and 4 in the project handout and section III in this report for more information.

While in automatic mode, once the sensor data is collected, a rule-based algorithm is followed to decide whether the pumps of zone 1 and 2 should be turned on or off. Pumps will be turned on only if the soil humidity drops below a certain level at a given temperature range. However, soil pH level has been prioritized as a soil that is too acidic or too alkaline could kill the plants. If the soil pH level is not within an acceptable range, water will be pumped to the plant regardless of the air temperature or soil humidity. If the temperature drops below 5 °C, a red LED will flash every 1.5 seconds to indicate to the user that frosting is occurring and subsequently both pumps will be turned off. If the temperature exceeds 50 °C, the LCD will display a message on the first line of the LCD alerting the user to check for faulty sensors and both pumps will be turned off subsequently.

The code was decomposed into several subroutines. Table 1 shows a brief description of the most important ones, while Figure 1 demonstrates the overall system flow in the form of a flowchart. Be aware that these two do not tell the whole story but are supposed to give the reader a sense of what is happening. It is highly advised to check the .asm file in order to grasp the inner-workings of the system in full.

Table 1: *Brief Description of the Main Subroutines*

INITIAL	**LOOP**
Initializes registers, configures the A/D, LCD, interrupts, Timer0 and ports.	Checks if the system is in automatic or manual mode to decide to go to either **AUTO** subroutine or **MANUAL** subroutine.
MANUAL	**AUTO**
Displays 'MANUAL MODE' on the LCD and goes to **AUTO_WORK** subroutine which after calling all subroutines below this row goes back to **LOOP** subroutine.	Displays 'AUTO MODE' on the LCD and continues to **AUTO_WORK** subroutine which after calling all subroutines below this row goes back to **LOOP** subroutine.
CONVERT_TEMPERATURE	**CONVERT_HUMIDITY1 and CONVERT_HUMIDITY2**
Converts the analog air temperature reading to a digital one using the A/D module, adjusts the reading to become a value ranging from 0-100, checks if frosting or a faulty sensor were detected by calling subroutines **FROST** and **FAULT** and finally calls subroutine **DISPLAY_TEMP** to display the air temperature on the LCD.	Converts the analog soil humidity reading to a digital one using the A/D module, adjusts the reading to become a value ranging from 0-100 and calls subroutine **DISPLAY_HUMIDITY1** or **DISPLAY_HUMIDITY2** to display the soil humidity on the LCD.
CONVERT_PH1 and CONVERT_PH2	**CHECK_FOR_FROST**
Converts the analog soil pH level reading to a digital one using the A/D module, adjusts the reading to become a value ranging from 0-31, subtracts one if the value is not zero effectively changing the range of values to 0-30, checks if a half should be added to the reading after dividing by two and calls subroutine **DISPLAY_PH**1 or **DISPLAY_PH2** to display the soil pH level on the LCD.	If frosting is happening and the system is in automatic mode, both pumps will be turned off, but if the system is in manual mode nothing will happen.
CHECK_FOR_FAULT	**RBA1 and RBA2**
If a faulty sensor is detected and the system is in automatic mode, both pumps will be turned off and 'FAULT ALERT' will be displayed on the first line of the LCD, but if the system is in manual mode the message will be displayed on the LCD but both pumps will be left untouched.	If the system is in manual mode, or if frosting is happening or if a faulty sensor is detected, nothing will happen. Otherwise, the control algorithm will be followed to decide whether pump 1 or pump 2 should be turned on or off while prioritizing soil pH level reading.

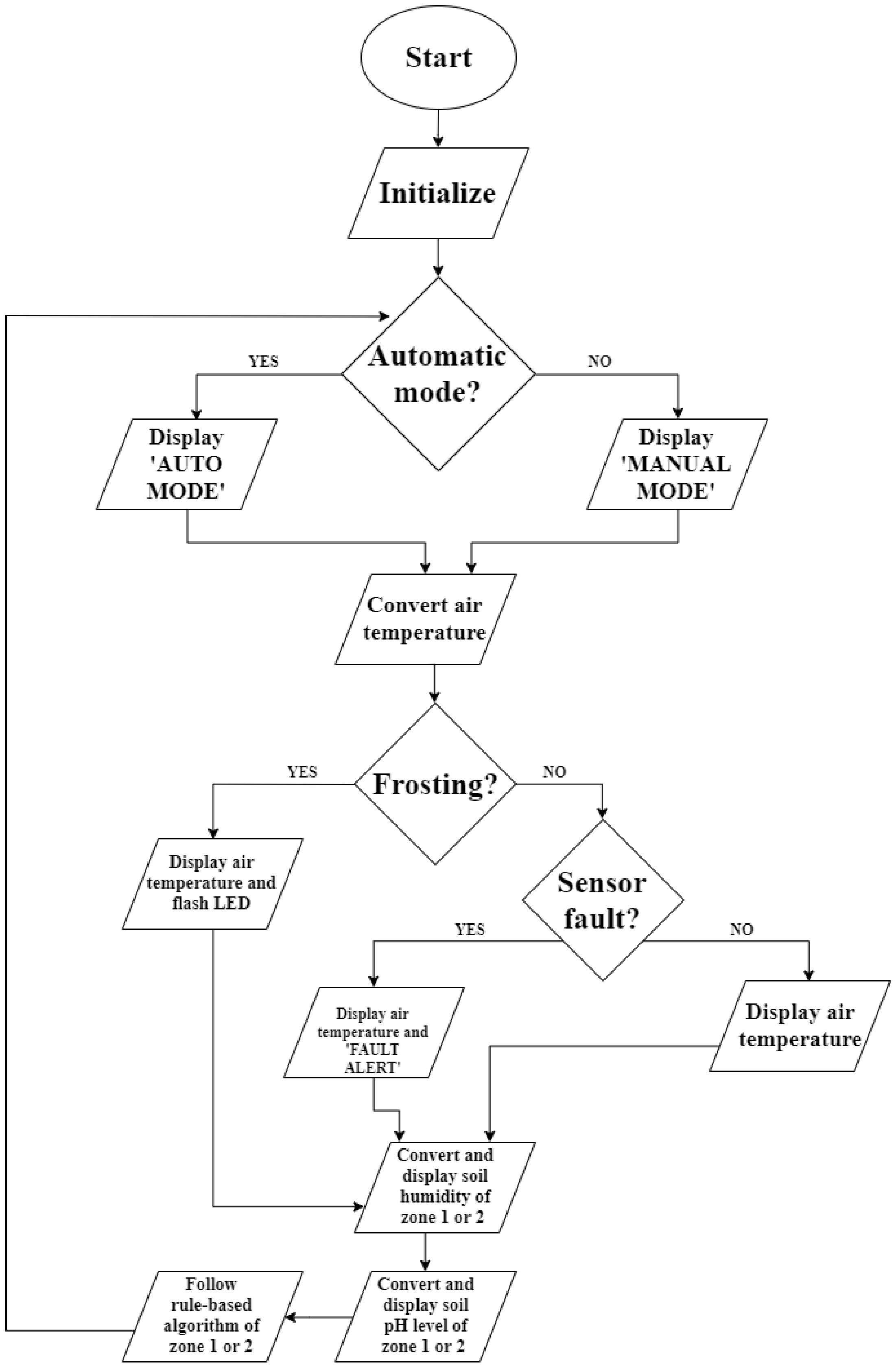
Start
Initialize
Automatic mode?
YES
NO
Display 'AUTO MODE'
Display 'MANUAL MODE'
Convert air temperature
Frosting?
YES
NO
Display air temperature and flash LED
Sensor fault?
YES
NO
Display air temperature and 'FAULT ALERT'
Display air temperature
Convert and display soil humidity of zone 1 or 2
Convert and display soil pH level of zone 1 or 2
Follow rule-based algorithm of zone 1 or 2

Figure 1: *Overall System Flow*

III. Hardware System

As stated before, the system has five sensors (potentiometers): one for air temperature, two for soil pH level, and two for soil humidity. Their outputs were connected to pins RA5, RA1, RA3, RA0 and RA2, respectively. The five pins were configured as **inputs to the PIC**.

The A/D module was initially configured with clock conversion of $F_{OSC}/8$, five analog inputs (AN0, AN1, AN2, AN3 and AN4) with V_{DD} and V_{SS} as voltage references, left justified result and AN4 (RA5) as the selected channel of conversion to convert the air temperature sensor reading. Then, channels AN0 (RA0), AN2 (RA2), AN1 (RA1), AN3 (RA3) and finally AN4 (RA5) were selected sequentially to continuously convert all readings into digital data.

The air temperature and soil humidity sensors transfer functions were initially realized the same way. By inspection, we find that a change of 1 °C in air temperature or 1% in soil humidity translates into a change in voltage by 0.05. The input voltage range of the A/D module is 5 V (5 - 0) which means that each value of digital output represents an interval of 4.88 mV (5/1024). From this, we can deduce that when the air temperature or soil humidity changes by 1 °C or 1%, the output voltage of the sensor changes by 4.88 mV which causes the A/D digital output to change by 10 (0.05 V/4.88 mV). This explains why in the code, the value in register ADRESH was divided by 10. The same procedure could have been applied to the pH sensors, but for the sake of accuracy, we decided not to follow it. Instead, the value in register ADRESH was divided by 8.

The Timer0 module caused an interrupt every 10 ms by having it configured to increment through internal instruction cycle clock, with a prescaler of value 32 and initializing TMR0 register with a value of 178. Another register, COUNTER, was introduced into the ISR code with a value of 150. Every time this register hit zero, the red, 'frost alert' LED changed state from on to off or vice versa, causing it to flash every 1.5 seconds in case frosting is occurring.

An LCD (QAPASS 1602A) is connected to the system. We opted to use the eight pins of PORTD, configured as **outputs from the PIC** to send commands to the LCD. Additionally, pins RS, RW and E of the LCD were connected to pins RE0, RE1 and RE2, respectively, which were also configured as **outputs from the PIC**. The V_0 pin of the LCD was connected to ground as we were not planning on changing the LCD's contrast.

Four push-buttons are also connected to the system. One to change which zone is having its data displayed on the LCD, one to control pump 2 in manual mode, another to control pump 1 in manual mode and one to change mode from automatic to manual or vice versa. Pins RB4, RB5, RB6 and RB7 were defined as **inputs to the PIC** to make use of PORTB change interrupt and the four push-buttons mentioned above were connected to them, respectively. The four push-buttons were interfaced using four, 10 kΩ resistors, which means they normally give logic zero and when pressed down give logic one (normally open 'NO'). Software de-bouncing was used.

The pump 1 LED, pump 2 LED and frost alert LED were connected to pins RC0, RC1 and RC2, respectively. The three pins were defined as **outputs from the PIC**. The three LED's were interfaced as current-source using three, 1 kΩ resistors, meaning they will emit light when the PIC outputs logic one, and stop when the PIC outputs logic zero.

A crystal oscillator of frequency 4 MHz was connected to OSC1 and OSC2 pins. Two loading, 33 pF capacitors were used to get stable oscillation from the oscillator.

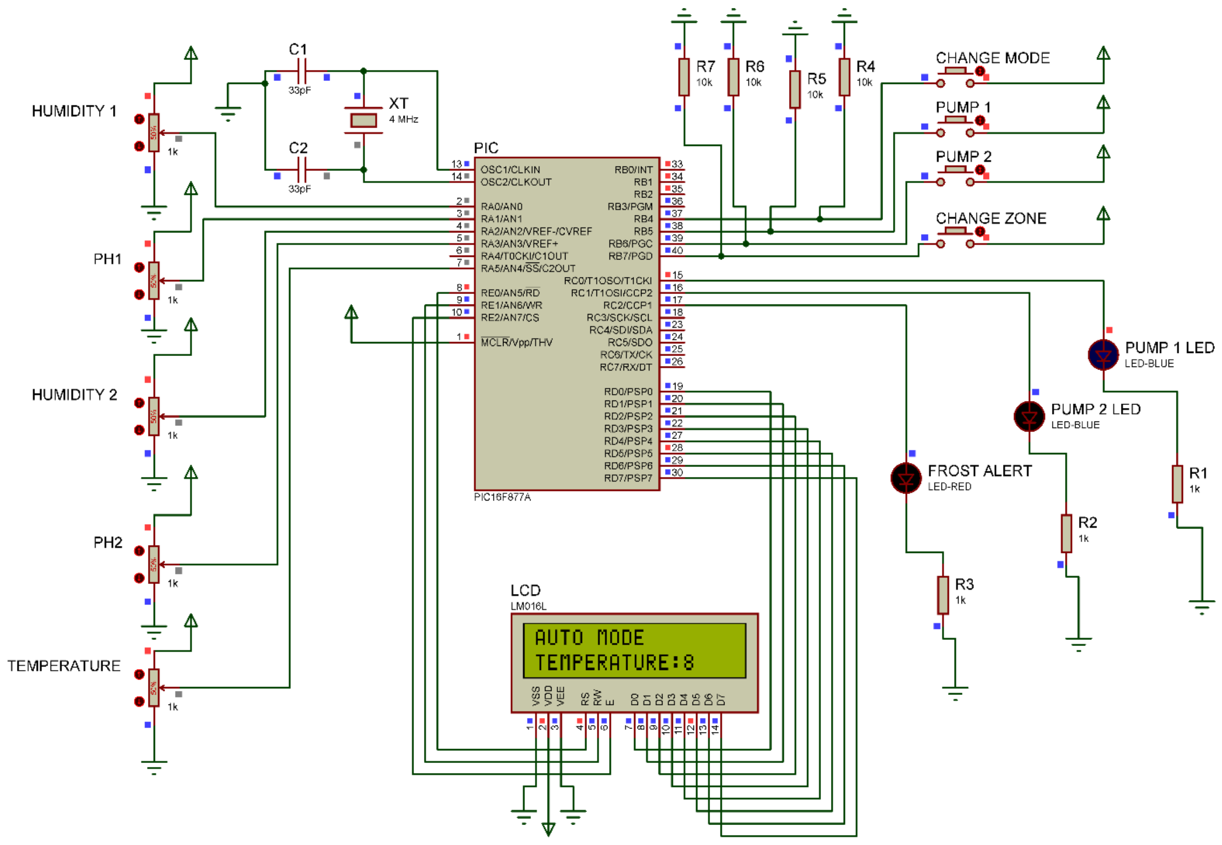

Figure 2: *Circuit Design Schematic*

Figure 2 above shows the circuit design schematic.

IV. System Testing and Results

a. Unit Testing

All subroutines were tested as part of the main code using the MPLAB debugger, while some were isolated and tested on an individual basis. Different inputs were tried to test the validity of the output(s) of the subroutine and the main code. Clear naming conventions (labels) were followed to avoid confusion. We also made sure the results of unit tests did not change regardless of whether we wanted to make improvements to our subroutine or if the requirements of what the subroutine should do changed.

b. System Integration

A hybrid integration approach was used. We started out with top-down integration by creating a main subroutine which controls the operation of the entire system. Then, bottom-up integration was adopted to add the other subroutines one-by-one. For example, after writing the **AUTO_WORK** subroutine, subroutines **CHECK_FOR_FROST** and **CHECK_FOR_FAULT** were written and added into the mix (top-down integration). Another example is subroutines **SEND_CMD** and **SEND_CHAR**, which were written before implementing the subroutines necessary to display messages and readings (bottom-up integration).

c. Overall Test

Table 2 demonstrates the test cases that were chronologically used, their expected result and whether or not that result was met. The testing was done using Proteus.

Our choices were made to consider all cases in which something out of the norm happened, e.g.: frosting was happening, faulty sensors were detected, the user tried to turn pumps 1 and 2 on or off while the system was in automatic mode. On top of this, several cases from the two control tables were tested to see if automatic mode works as intended or not (it did). This should cover all possible regions of operation of the system, although there probably still exists a combination of sudden changes in inputs that overwhelm the system and cause it to malfunction. This will not be considered since in real-life, changes in soil pH level, soil humidity and air temperature are gradual and seldom sudden.

Table 2: *Overall Test*

Test Case	Expected Result	Actual Result
Starting the system.	System should be in automatic mode and display stats for zone 1.	Just as expected.
Changing temperature to below 5 °C.	Frost alert LED should flash every 1.5 seconds and both pumps should be turned off.	Just as expected.
Changing temperature to above 50 °C.	Frost alert LED should be turned off and both pumps should be turned off.	Just as expected.
Changing mode to manual during frosting and faulty sensor cases and trying to turn both pumps on and off.	Alert for either should still be displayed but pumps should turn on and off.	Just as expected.
Changing which zone will have its' data displayed on the LCD.	Data for the corresponding zone should be displayed on the LCD.	Just as expected.
Trying to turn both pumps on and off while system is in automatic mode.	Nothing should happen.	Just as expected.
Changing values of the five sensors to see if the rule-based algorithm is followed.	Pumps should be turned on or off based on the values of the five sensors.	Just as expected.

V. Conclusion

In this project, we managed to design a simple automated home garden irrigation system. The main code was split into several subroutines. Unit testing was undertaken to test subroutines before finally adding them to the main code. A hybrid integration approach was adopted to put everything together. Edge test cases were simulated to check the validity of the system's outputs. No major obstacles were faced during the design process. Table 3 below roughly shows the contribution of each student.

Table 3: *The Contribution of Each Student*

Task	Student(s)
ADC code	Yusuf
LCD code	Osama
Rule-based algorithm code	Yusuf and Osama
Writing report	Yusuf
Buying hardware	Osama
Assembling hardware	Yusuf
Documenting code	Yusuf