ECE36200 Final Lab Project Report

Smart-Self Sustaining Garden

Team Members: Stephen H. Corya, Andrew A. Lorenz

April 26th, 2024

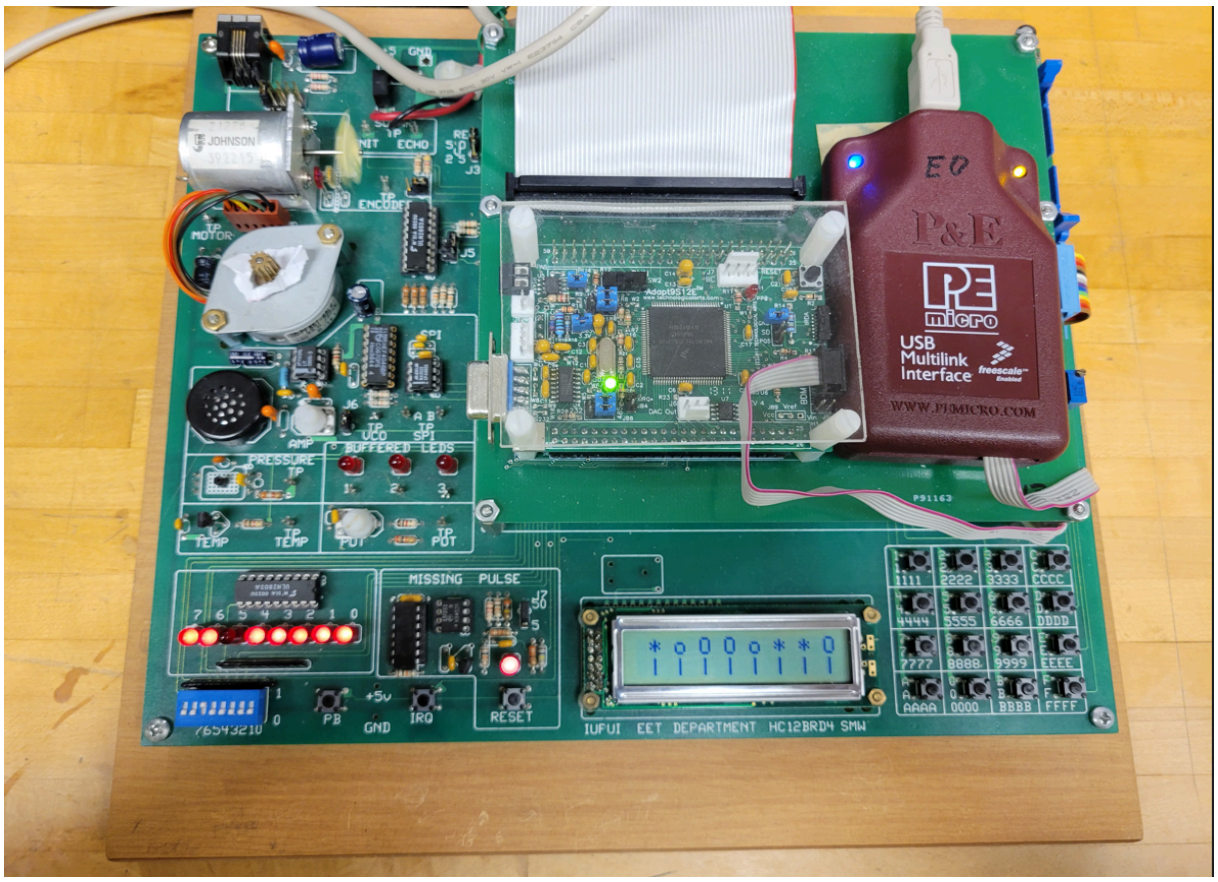


Table of Contents:

List of Tables and Figures:

Introduction:

The Smart Self-Sustaining Garden is a simple concept that is easy to understand just from observing the title. The purpose here is to simulate the core aspects of a working garden that can function on its own with limited, but necessary, user input. This will be achieved by creating an HC12 assembly language code to control a MC9S12E128 microcontroller. This device in the Lab will give all the needed functionality for simulating the garden. The device will simulate details including: generation of the layout of the garden, temperature adjustments, controlling LEDs, a watering system, and displaying the time and date.

Design and Purpose:

To begin, the used I/O board features will be described here. First, the LCD screen, located on the lowest portion of the board, serves the most basic purpose. This purpose is to display our menu options, show valuable data about the garden, and inform the user the action is being performed. Next, the hex-keypad, on the bottom right corner, is another important part of the design in the project for the purpose of performing the menu actions, entering the password, and entering data about the time and date. For the simplicity of the design, we did not use the buttons 8-F for any of the actions except for the password. Third, we used the LEDs located on the bottom left corner of the board. These LEDs only light up when the correct password is entered. However, the buttons on the keypad do not use the LEDs but instead on the switches lower down. Next, above the LEDs is the potentiometer. This is a dial that controls the speed of the DC motor further up on the board near the left corner, reflecting the fan used in a garden. This is related to the LCD screen because, depending how fast the motor is spinning, the LCD will display the temperature in the garden. Below the DC motor, is the stepper motor, which is used like a sprinkler in a garden when we input the button to water the garden. These are all the important features of the board we have used in the project.

Hardware implementation of the Smart Garden:

This is how the hardware will function when the project code runs. When the program begins, the LCD screen will display “Password:” and we are able to enter any password we want since we were unable to hardcode the password. Once that is done, the DC motor will spin, and the LCD board will display the menu. Each option is shown for 2 seconds before displaying the next one and cycles back once the last option is shown. There are 7 actions that can be performed by pressing one of the buttons on the keypad.

1 is the watering action which will activate the stepper motor. The motor will rotate two steps forward and one step back on each interval. The stepper motor only runs for about 4 seconds until it stops as intended. The LCD screen, at the same time, will display “watering…” and then will return to cycling through the menu options when it is done.

2 is the “show temperature action which will display the temperature of the garden. The potentiometer is the dial that will change the speed of the DC motor. The DC motor will spin and and stop for the second and spin again when is is used. When the potentiometer is turned all the way down, the motor stops completely, and the LCD screen will display it’s lowest temperature of 66 degrees. When is is turned up entirely, the dc motor will constantly spin, and the lCD screen will show 80 degrees.

3 is “show time”. This will display our default time of 01/01/1970. This is a constantly running clock but it is significantly slower than the normal clocks used everyday. This time and date can

be changed manually through the “change time” action for button 4. When we press that button, the LCD board will show the prompt to enter our password. We then proceed to manually enter in the time and date. When that action is completed, we press the 3rd button and we see that it is exactly the time we entered before.

5 is for resetting the password. This allows us to enter a different password once the old password is entered again.

6 is our planting feature. This is is to place plant in our garden and when we press number 7 on our keypad afterwards, the LCD screen will show the characters that imitate the visual of a garden. If 6 is not pressed, the garden will be empty.

Software Implementation of the Smart Garden:

The general software controls for this project are handled by the real-time interrupt routine, which we have called an “event loop.” The event loop runs every four milliseconds. In general, the event loop sets flags which are then read by the subroutines called in the main.asm program. The subroutines called by the main.asm file run with much greater frequency than the control flags set in the event loop. Because the event loop runs every four milliseconds and the main loop runs at the full speed of the microcontroller’s clock, flags set in the event loop are read (for our purposes) immediately by the subroutines called in the main loop.

The Main Loop:

Within the file main.asm are all of the variables and flags that will be used by the subroutines as well as the event loop. The main loop calls subroutines Password, Menu, Circulation, Water, Growlights, Keypad, and Wall (Clock). These subroutines are run indiscriminately in the main loop. The controls and conditions for their executions are set via flags from the event loop.

The Event Loop:

The “event loop,” defined in eventloop.asm, is the RTI loop. It runs at a four millisecond interval. The event loop handles the bulk of user interface controls with its timers and flags.

General Flow:

Upon the initial start of the program, the user is prompted for a password. No other subroutines are called until a password is entered, and the RTI loop exits without running any subsequent steps nor setting any flags, the wall clock flag indicating a second has passed notwithstanding.

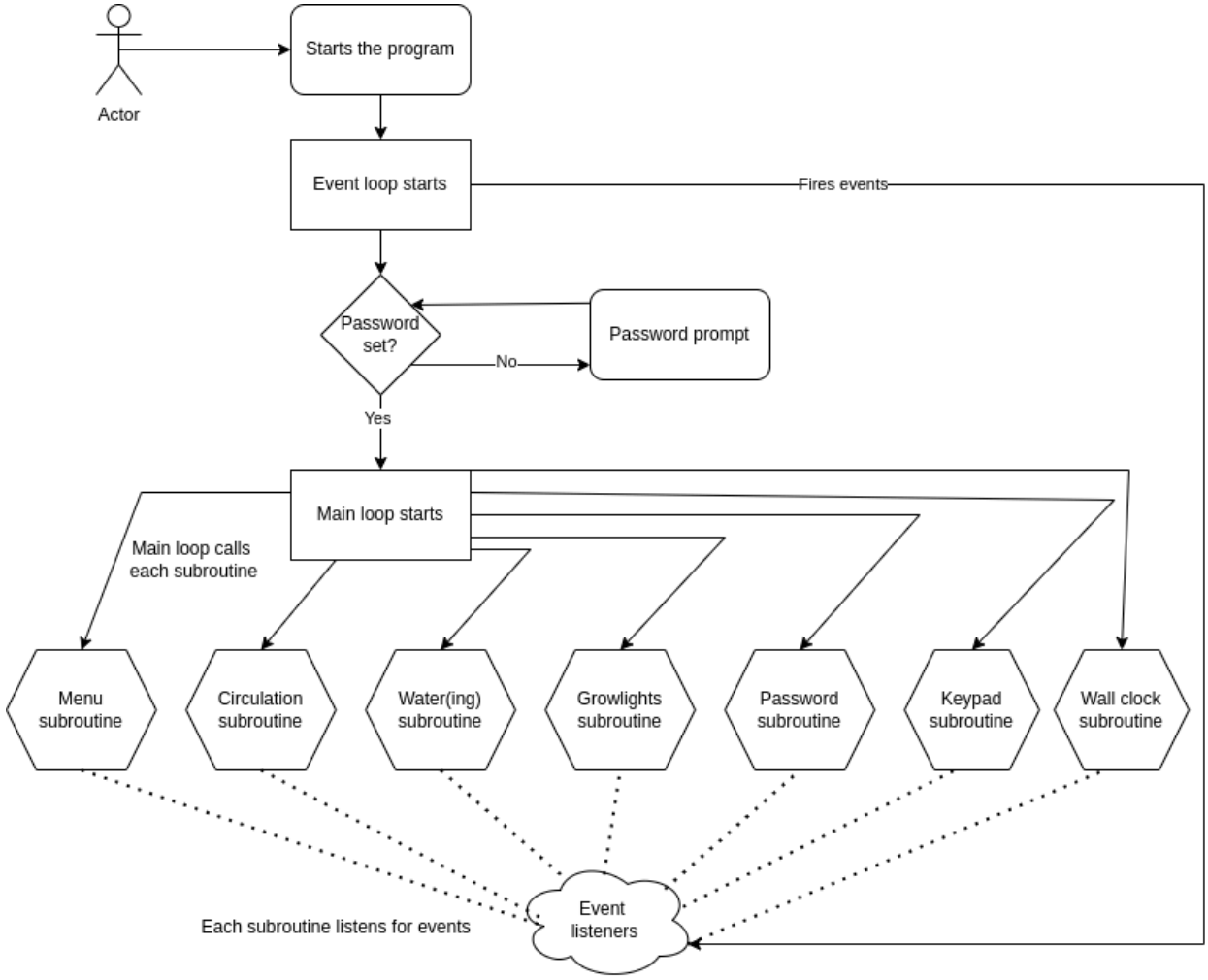


Figure 5.b.1: General Program Flow

After password initialization, the two loops (the event loop and the main loop) begin to coordinate with one another to run the various functions of the software. The main user input device, the Keypad, is called as a subroutine within the main loop. The main loop listens for keypresses, which are in turn read by the event loop.

Given that the event loop runs only every four milliseconds and the main loop runs as fast as the microcontroller's onboard clock permits, one may find it prudent to consider the events from the main loop as happening before events from the event loop.

Now that the general flow of the software features has been outlined, let us further examine how the two loops communicate with one another. Subsequent figures within this section will assume that the password has been initialized, and that the event loop and the main loop are both running.

Within the main.asm file, a variety of bytes and words are defined and initialized. Many of these are event flags. These event flags are generally set within the event loop, to be read within the main loop. Oftentimes, when an event is read by the main loop, a subroutine will act on this event, then change the event flag. This is not the case with the keypad; the main loop reads the keypress, sets its value in the appropriate variable, and then it is read within the event loop. In the case of the keypad, a variable is set within main, read by the event loop, then changed upon the completion of some instructions in the event loop.

The Keypad Subroutine:

The keypad was extended from the versions in the previous laboratory projects to allow for configuration. By setting variables, which are defined in the main.asm file, the keypad can be configured to await or not to await a keypress, as well as to await or not to await a keyrelease.

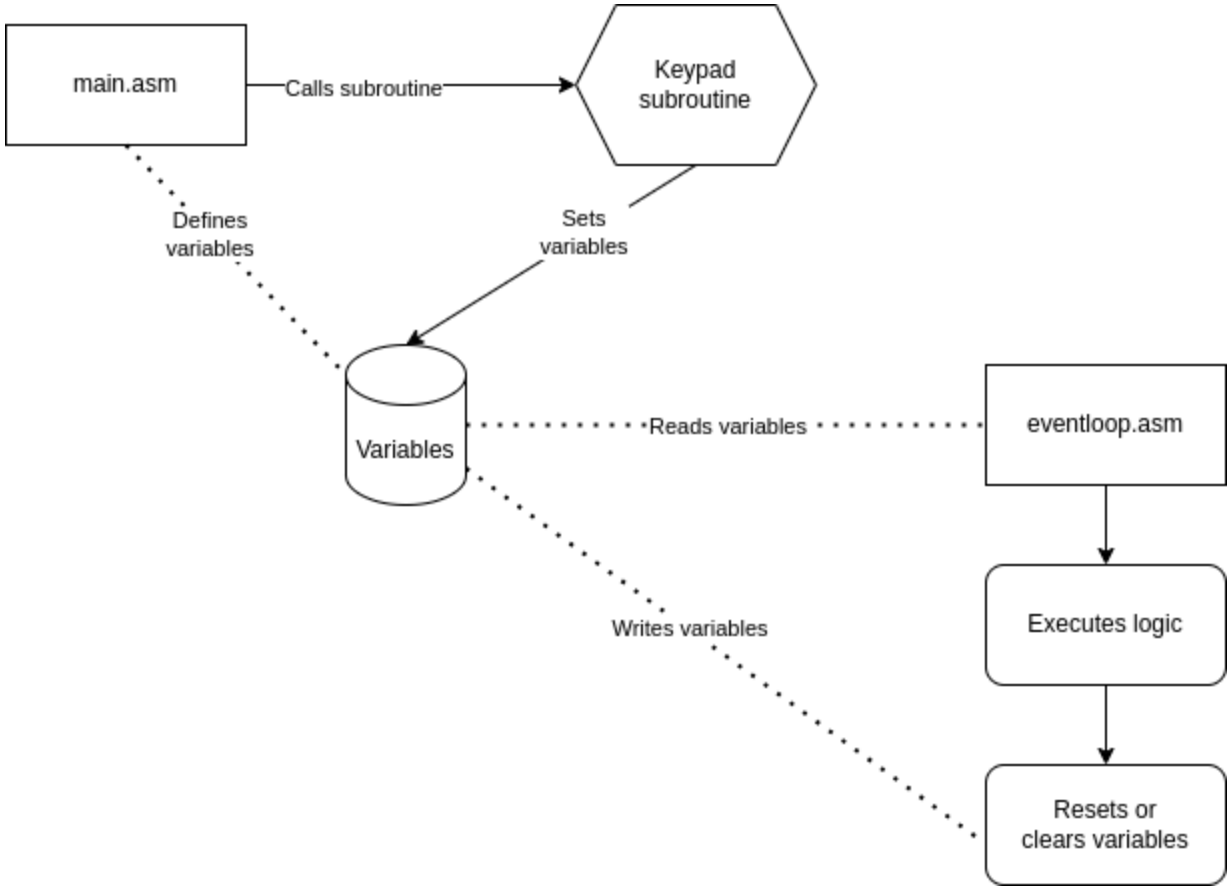


Figure 5.b.2: Keypad Subroutine Flow

Depending on the values read from the Keypad subroutine, the event loop sets several state variables which define how the other subroutines in the main loop will run. In general, the event loop resets these variables after altering the other state variables.

The Menu Subroutine:

This subroutine is responsible for the primary user interface: a menu that shows which keypress will trigger one of the program's functions. The menu cycles through the user's options, displaying one option at a time. Each option has a corresponding number, which in turn corresponds to a keypad number. The menu cycling functionality is set by a timer in the event loop. When this timer reaches its predefined limit, a variable controlling which menu option is displayed is incremented. The variable is reset within the Menu subroutine after the final option has been presented to the user, and the Menu again displays the first option.

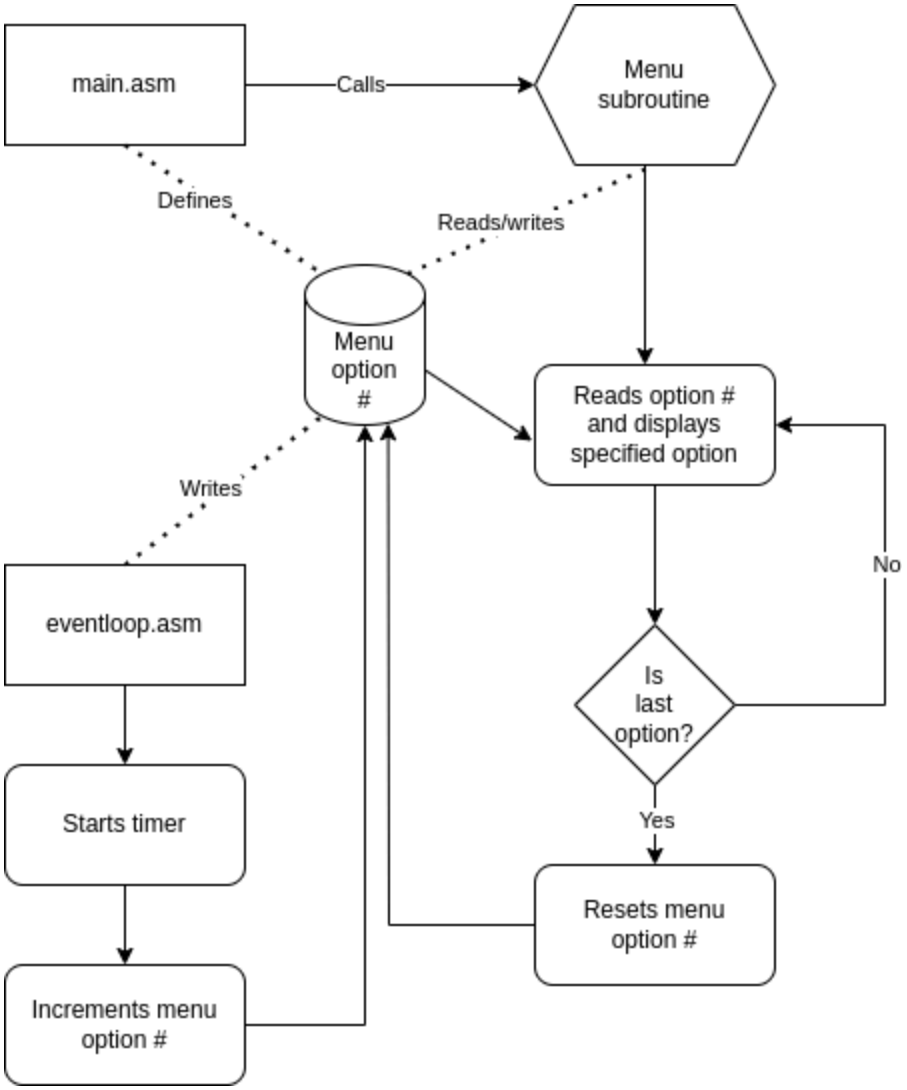


Figure 5.b.3: Menu Subroutine Flow

The Watering Subroutine:

The watering function is the first menu option presented to the user. Upon the press of the “1” button, the event loop, on its next execution, reads this input and reacts by setting the state variable which causes the Watering subroutine to run; simultaneously, the menu is hidden.

In addition to the variable which defines whether or not the garden is in its watering state, another state variable controls the sprinkler (the stepper motor.) The sprinkler variable is a simple flag that is set within the event loop, then cleared within the Watering subroutine, after the motor has moved. Note: the sprinkler may have run more smoothly if its event flag was both set and cleared within the event loop.

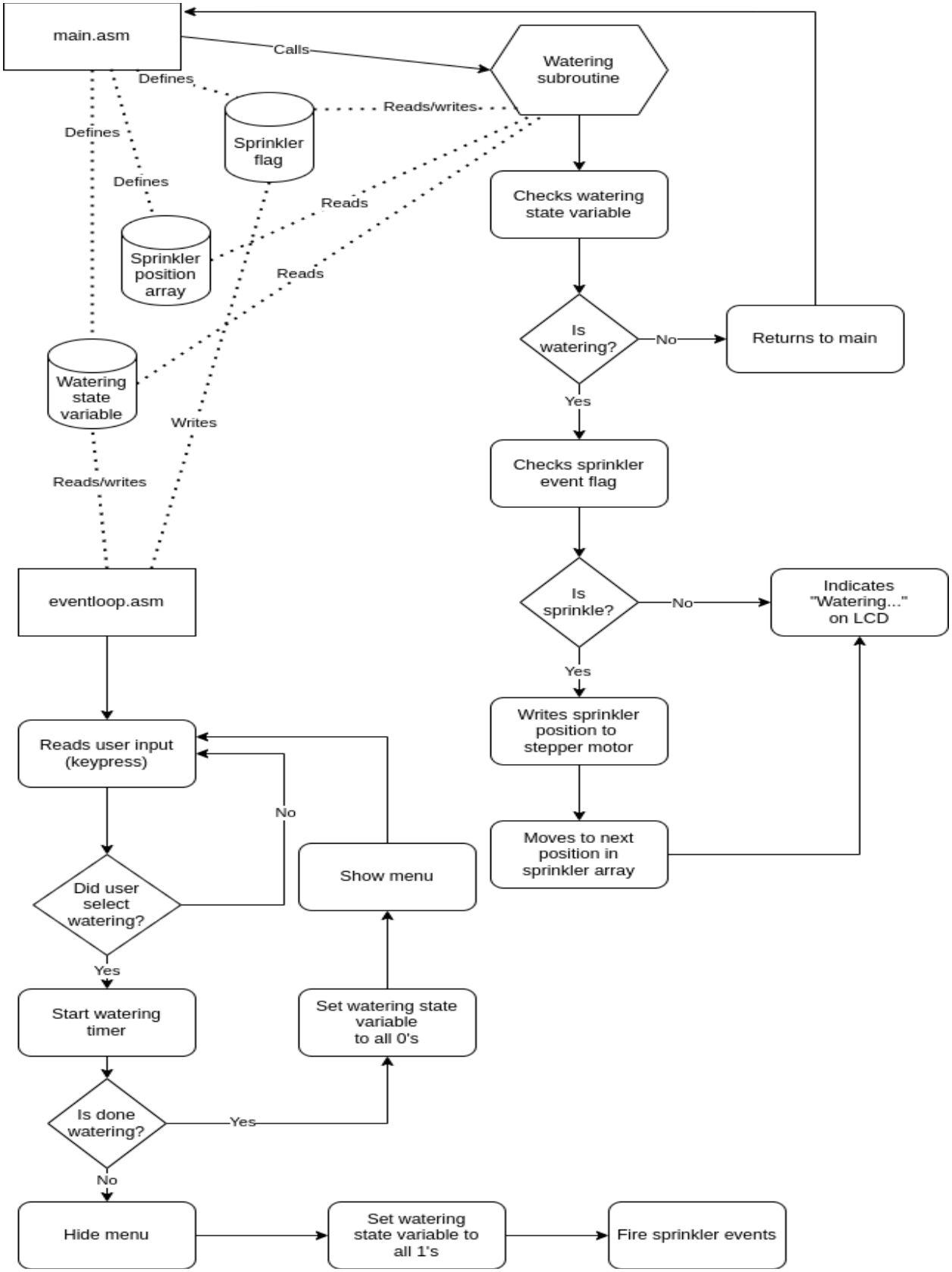


Figure 5.b.4: Watering Subroutine Flow

The Password Subroutine:

The Password subroutine handles two different password operations: initialization and reset. The general program does not run any subroutines or fire any events (other than the wall clock) until the password is initialized. Initialization is controlled by a state variable, which is unset until the password is successfully read for the first time. A password consists of eight characters. Any password that can be input on the keypad can be set in the initialization phase. The password prompt displays underscores for characters that have not been set, and asterisks for characters that have. This provides the user with a visual representation of how many characters he has entered so far. Upon initialization, the password is stored in RAM.

Similar to initializing the password, resetting the password is controlled by a state variable. This variable is set to all 1's when the user has selected the option to reset his password. The user will be prompted for his current password. If he does not enter it correctly, he will be prompted again. A feedback interface with underscores and asterisks is provided to the user in a manner much the same as when he initializes his password. Resetting the password is a synchronous operation; it blocks other input and output until the user enters his entire password. Upon successful entry of the current password, the user will be prompted to enter a new password.

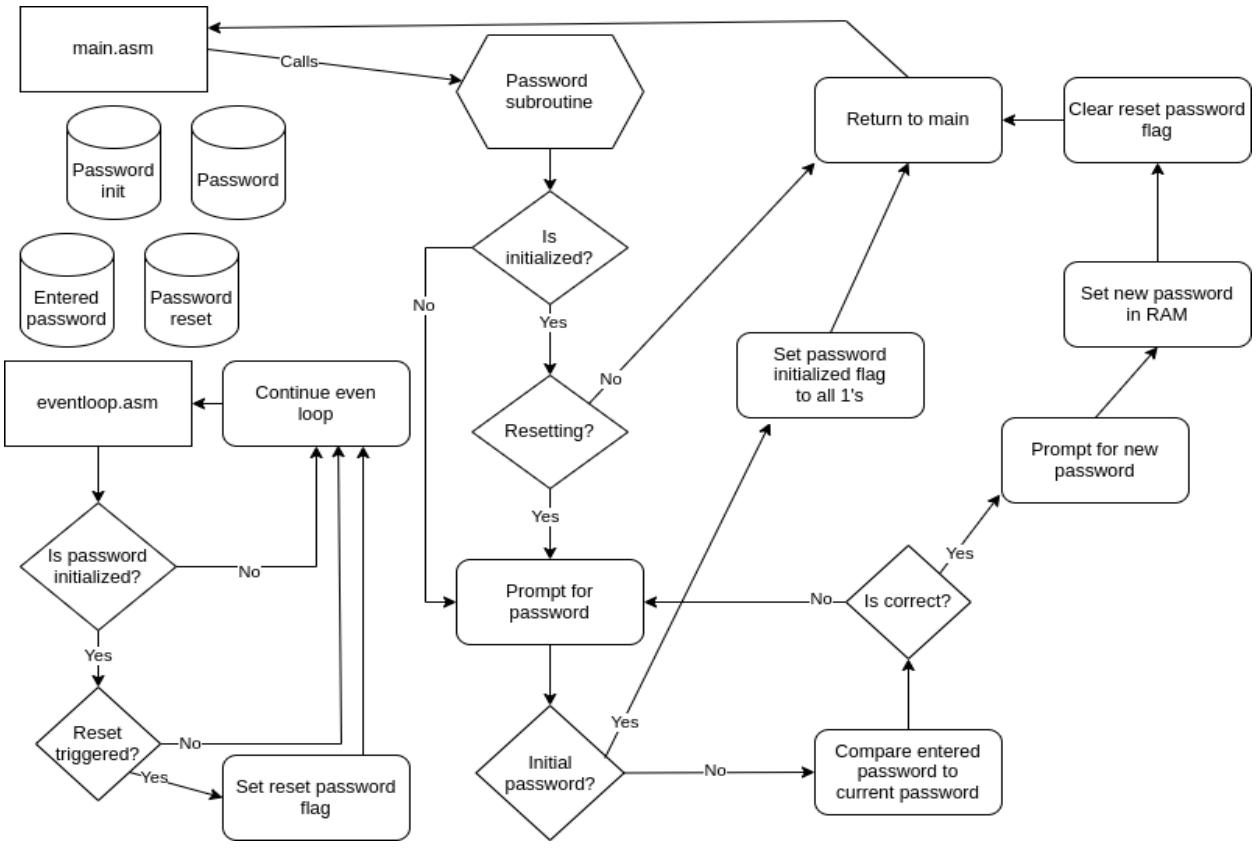


Figure 5.b.5: Password Subroutine Flow

The Circulation Subroutine:

The Circulation subroutine spins the circulation fan (DC motor) and prints the temperature to the LCD screen. The calculation of the temperature is such that the temperature increases as the fan speed decreases, and both of these variables are set according to the value read from the potentiometer. The potentiometer is read within the event loop, and the calculation of both the fan speed and the temperature are handled by the event loop. The event loop sets a flag that is read by the Circulation subroutine and which outputs a pulse-width modulated signal to the DC motor.

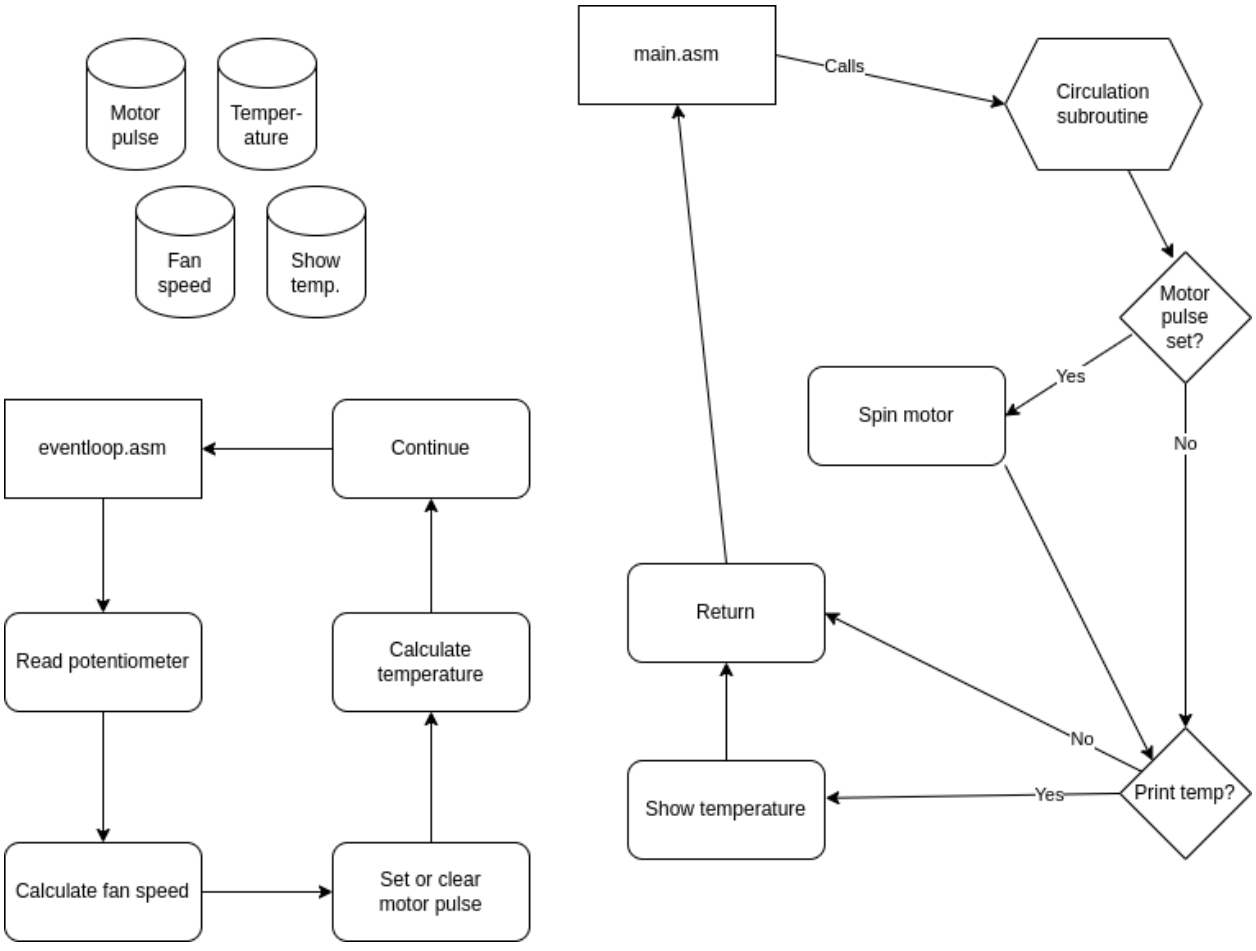


Figure 5.b.6: Circulation Subroutine Flow

The Wall Clock Subroutine:

The Wall Clock subroutine, sometimes referred to as “Wall” outputs the real-world time. It defaults to one of the programmers’ birthday. The clock has been programmed such that there are exactly thirty days for every month, so there are only three-hundred sixty days in the year. We use the American date format of “month/day/year” and twenty four hour time. The main loop calls the Wall subroutine, and the event loop fires an event with every second passed. This is cleared after it is detected and acted upon in the Wall subroutine. The Wall subroutine counts these seconds, and with those values, increments the minute, hour, day, etc.

There are two menu options for the wall clock: set and display.

Setting the clock is similar to setting or resetting the password, and it requires the correct password to be input before the time can be set. The means of checking the password before setting the date and time are the same as those used when resetting the password. When setting the date and time, the values yet to be input have their places held by underscores on the LCD display.

Within the setting function, the user is only able to enter valid values. That is, it is not possible to enter “12/32/1993” or “13/01/1995” as the current date. If the user attempts to enter an invalid value, the underscore is not replaced by the number input by the user, and the user should again attempt to input a valid value.

Showing the clock requires the conversion of the values of the current month, day, year, etc. values to ASCII. This is done using division by tens, hundreds, and thousands, and using the quotient and reminder from these division instructions.

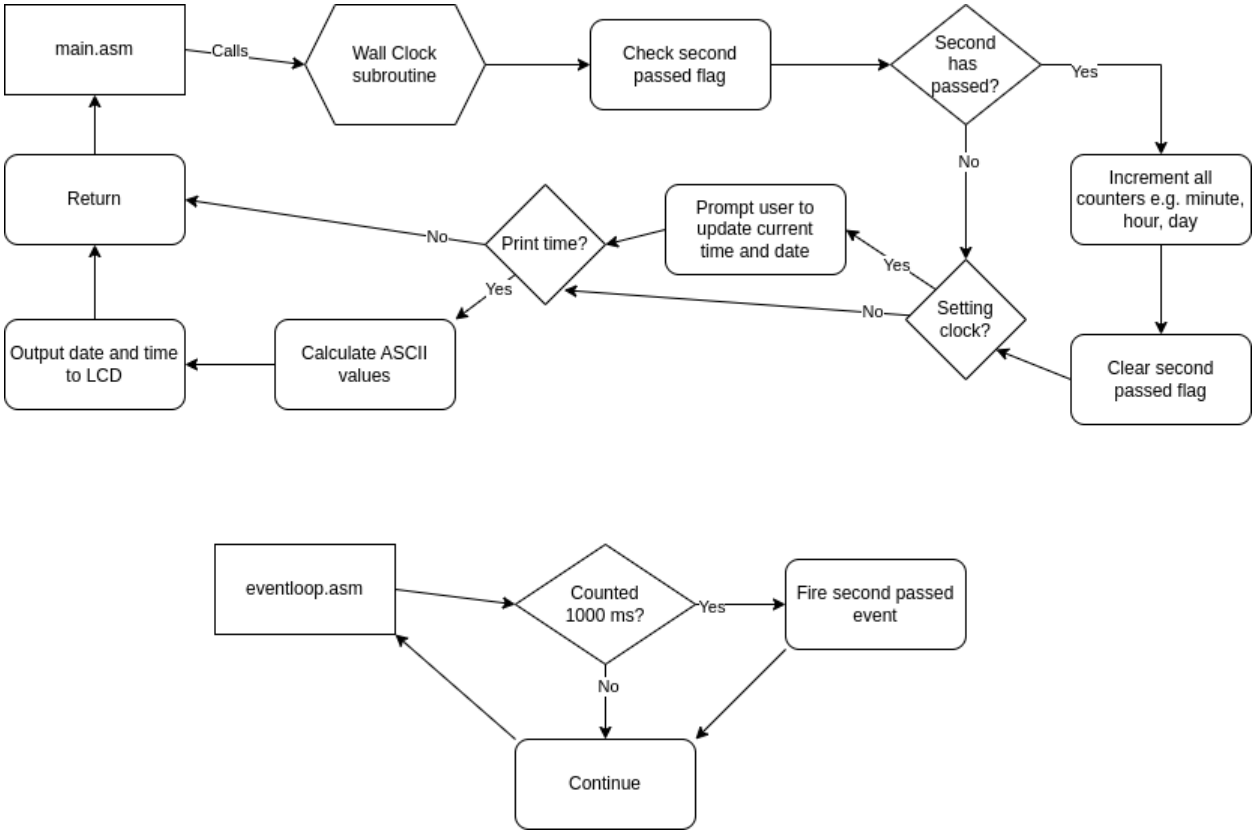


Figure 5.b.7: Wall Clock Subroutine Flow

The Growligts Subroutine:

Owing to the simplicity of this subroutine, a flow may be regarded as superfluous. The subroutine simply pushes the various registers and accumulators to preserve them atop the stack, reads the DIP switches into Accumulator A, writes this value to the LED's, restores the registers and accumulators via pull instructions, then exits.

The Sow Subroutine:

The Sow subroutine displays "Planting…" on the LCD screen, and it sets the seeds planted flag to all ones.

The Plot Subroutine:

This subroutine checks if the seeds have been planted. If they have, it checks the month during which the seeds were sown. If the seeds were not planted in April or May, then the plot is empty; otherwise, a crude emulation of plants and flowers is shown on the LCD.

Error Handling and Fail Safe Techniques:

Within the event loop, many variables are set which control the state of the subroutines called in the main loop. These variables should be mutually exclusive, and many should not be defined simultaneously. For example, if the system is watering, the user should wait for the watering to conclude before viewing the temperature.

The flag that indicates whether or not the password has been set is checked not only in main.asm, but in many of the subroutines. Nothing should happen in the program until the password is set.

Extra features:

In addition to our core features, we added 2 simple features that enhance the user experience. The first feature includes password encryption. This is a common feature for all password systems to prevent others from easily seeing the password.

The second feature is the condition that the planting will only work outside of the monthly range from January to March. This is only for simulating that it is only realistic to start a plantation in the spring and summer.
For our distribution of work on this project, both of us fulfilled key roles for the success of the project.

Issues with the project outcome:

Though, the smart garden was certainly functioning reasonably well, there were a few features that were hindering the user experience. First of all, the safety feature of the alarm was not working at all. It would not run when we wanted it to.

Discussion and suggestions for future improvements:

The password entry was fast and responsive when setting the password initially, before any of the other code ran; however, subsequent entry of the password (and date/time) was significantly sluggish by comparison. The cause of this undesirable behavior could be explored.

One improvement to execution time could be done by firing a “screen refresh” event within the event loop and/or the Menu subroutine whenever the screen needs to update. This would prevent the screen from being refreshed on the LCD when none of the characters displayed were changed.

Another improvement could be made by not pulling or pushing registers that are not used within subroutines. In its current version, every subroutine pushes every register and accumulator, and then pulls each of those at the end. This being done with great rapidity within the main loop could be the cause of significant performance degradation.

Within the event loop, variables are loaded into registers unnecessarily. Each timing variable is loaded into Register X, incremented, stored into Register X, reloaded from Register X, and finally compared to its maximal value. It is not necessary to store and reload the timing variable from the register. Only the first load and the comparison are necessary.

User Manual:

This is the Smart Garden guide that will help anyone know how to use this smart garden on the I/O board. When you start this program. You will be prompted on the LCD screen to enter the password shown in this image here:

Figure 10.1

You can enter any password you prefer and it should be something easy to remember. Once that is done you are presented with a multitude of options each displaying on the LCD screen and cycling through one by one. These options are: 1. Water Plants, 2. Show Temp, 3. Show Time, 4. Set Time and Date, 5. Change Password, 6. Plant Seeds, and 7. Show Plot. Note: there is a DC motor located on the top left corner of the image that runs constantly after you enter the password. This is for imitating a fan in a garden.

When you press 1 on the Keypad, you will see the screen show watering and the stepper motor will spin a way that acts like sprinkler. If you hold down the button, it will continue to “water plants” indefinitely. Otherwise the screen will go back to the menu.

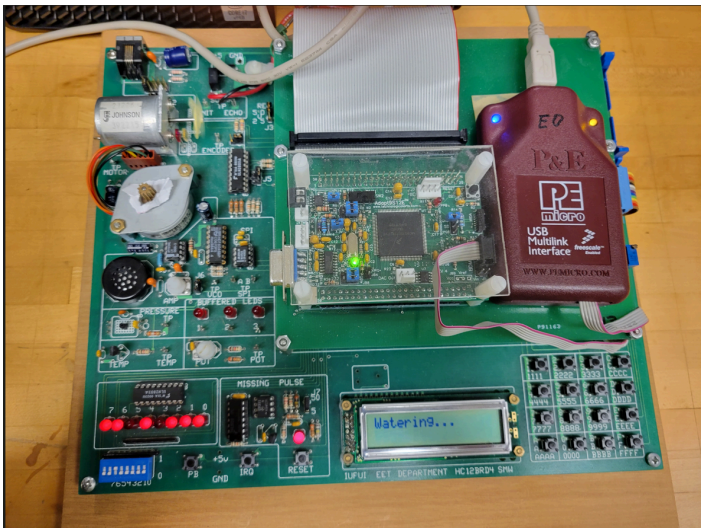
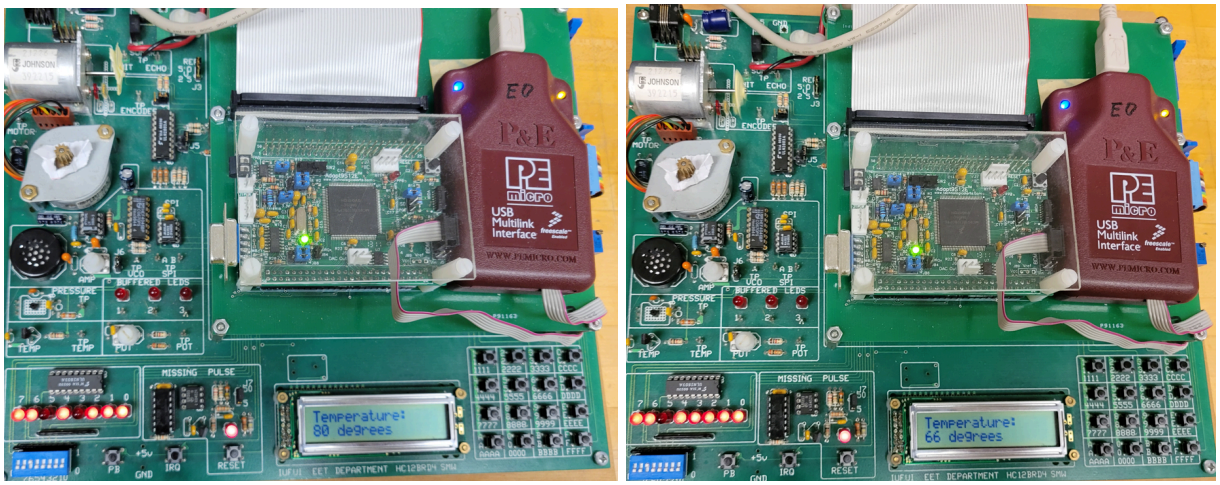


Figure 10.2

If you press button 2, you will be shown the temperature of the garden. There is a white dial on the board, above the 8 LEDs called the potentiometer, or POT for short. If you turn that, you will control how fast the DC motor will spin. Depending on the speed of this motor, the screen will display a range of temperatures from 66 to 80 degrees Fahrenheit. The motor will stop entirely if you turn down the dial all the way.



Figures 10.3 and 10.4

When you press button 3, this will show the default time set be the code of 01/01/1970. To change this time, press button 4 when the menu shows up again and you will be prompted to enter your password you made. After entering you password, you may now enter the current date and time of day on the screen with the keypad. Once that is done, the screen will display the menu again. Note: the date is a constantly running clock but it does not run quite as fast as the clocks we use everyday.

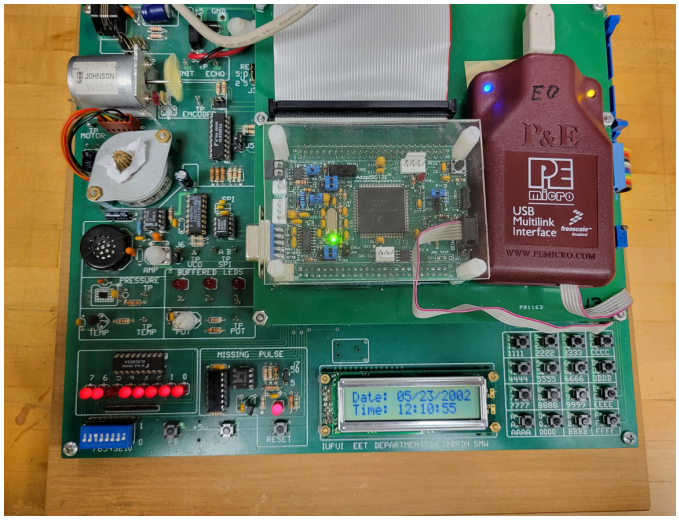


Figure 10.5

Next there is the feature that you may want to sometimes use, this being the change password feature. You can do this by pressing button 5 on the keypad. You will be prompted to enter your current password and only then will you be able to enter your new password. You will be returned to the menu after this.

Next you can simulate plantation with buttons 6 and 7. If you press button 7 first, it will show an empty plot. However, press button 6 and the screen will inform you the seeds are being sown. Press button 7 and you will see on the screen exactly like Figure 10.6. If you attempt to do this if you date and time are between January and April, the seeds will not grow and you will again see an empty plot. The plants will also disappear if you change the time to the monthly range of January to April after planting the seeds.

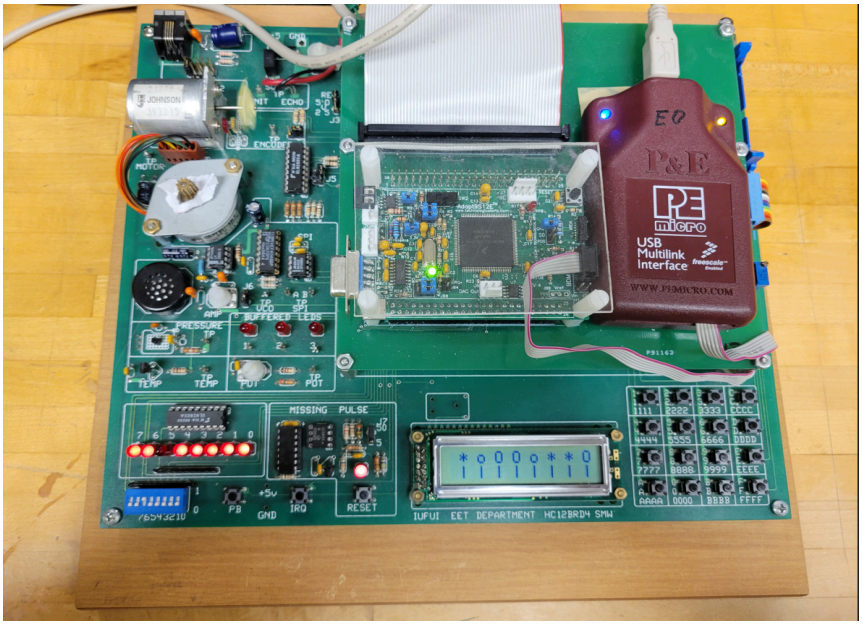


Figure 10.6

Hopefully, this user's guide was helpful for learning how to use this smart garden on an I/O board.

Conclusion

This project has been helpful in understanding two key concepts about creating a simple machine. First, through this project, we had come to understand how to create subroutines for many of the different features of the garden and applying this knowledge into this project. Second, the only true way to be able to hard code any machine, you must be able to have a deep understanding of writing assembly code if we want to be efficient with what the machine can do. This project was a challenge to understand and work over, but it helped to understand how we manufacture our machines.