The University of Jordan
School of Engineering
Department of Computer Engineering
Embedded Systems Course Project

A Simple Rain-Aware Automated Garden Irrigation System

Dr. Ashraf Suyyagh

Project Abstract and Main Requirements

In this project, you are required to implement an embedded system that will enable or disable the process of watering plants in a small home garden. The garden has two plant basins that we shall call *B1* and *B2*. The plants in each basin have different watering requirements, and we will sense the soil humidity to check if the soil is dry or humid to help us make better irrigation decisions. The system also detects if it is raining and therefore will stop the watering process so as not to saturate the plant. Some HMI (Human Machine Interfaces) are also present that we will introduce later. Figure 1 provides a drawing of such a system.

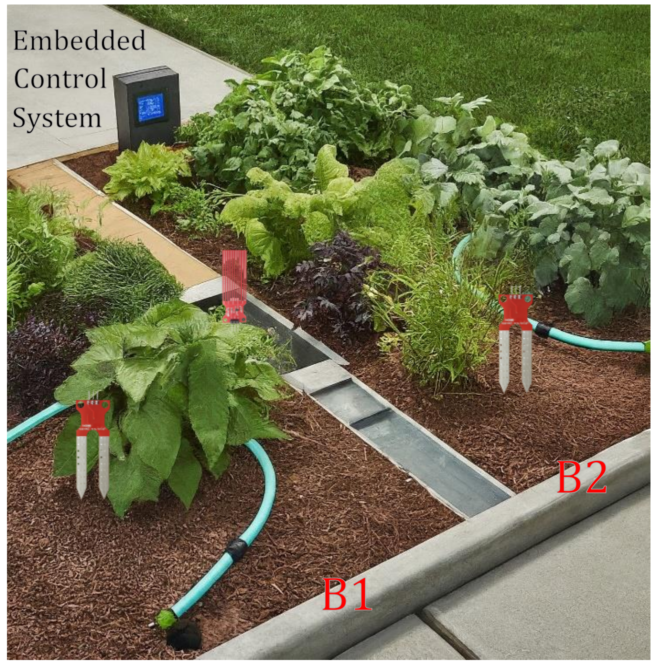

Figure 1 - Drawing featuring the garden and sensors.

Project Details

In each basin, there is a soil humidity sensor. The sensor is an analogue sensor that is powered by two pins V_{CC} and GND. It has one analogue output A_0. If the output is 0, this means the soil is completely dry. If the output is 5V, it means the soil has maximum moisture level (all soil pores are filled with water). Any value between 0V and 5V is proportional. So, if the sensor outputs 2.5V, it means the soil humidity is 50% of the maximum, and so on [1]. A drawing of the actual sensor can be seen in Figure 2.

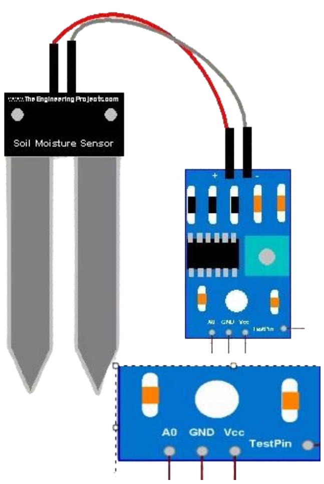

Figure 2 - Soil Humidity Sensor

Figure 3 - Water Level (Rainfall) Sensor

In the middle, and between the two basins, we have an electronic water sensor that we shall use to measure the level of the fallen rain. It detects the water by measuring the water's electrical conductivity. The sensor is made of 10 exposed vertical copper traces: five are power traces and five are conductive traces. The traces are placed in an alternating manner. When there is no water, then these traces are disconnected. When there is water, the water connects these traces. "*The exposed parallel traces work as a variable resistor whose resistance is directly related to the water level. The sensor resistance is inversely proportional to the water level. When the sensor is fully immersed it shows the lowest resistance, thus indicating more height of the water. And when the sensor is partially immersed, it shows more resistance, and less conductivity, thus indicating less height in correspondence with the resistance. This variable resistance is directly related to the voltage appearing across the sensor. By measuring that voltage, we can detect the water level*" [2]. Similar to the soil humidity sensor, the water sensor is powered by two pins V_{CC} (+) and GND (-). It has one analogue output S. **If the sensor outputs 0V, it means there is no water; therefore, it was not raining. The higher the voltage (up to 5V), means the higher level of rain fall.** Assume that the maximum water level this sensor can measure is 50mm (5 cm). A drawing of an actual sensor can be seen in Figure 3. An illustration of the relationship between the water level and sensor resistance is illustrated in Figure 4.

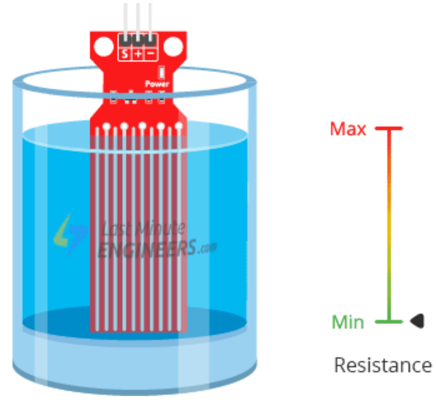

Figure 4 - Water Level to Resistance Relationship

Given that we have three analogue sources, we shall read them in order. We shall sample each source with a frequency of 1/3 Hz. That is, we shall sample each source every three seconds. So, we start with reading the water level sensor (rainfall), then after 1 second, we resume reading the B1 soil humidity sensor, then after 1 second, we continue reading the B2 soil humidity sensor and we repeat. This will effectively ensure the required sampling rate of 1/3 Hz per channel.

- You must use any of the HW timers with interrupts to generate the 1 second delay.
- You must use the ADC interrupt to know when the conversion is complete.

We will use a switch to choose if we want automatic or manual irrigation mode. If the switch is 0, this means we are in manual irrigation mode.

- Your rainfall sensor readings will also be interfaced to a LED barograph display to show the level of the water. The barograph is a set of LEDs as shown in Figure 3. The number of bars lit indicates a rainfall, each light represents 5mm.
- You should use two multiplexed 7-segment displays to show the percentage of the soil humifity.

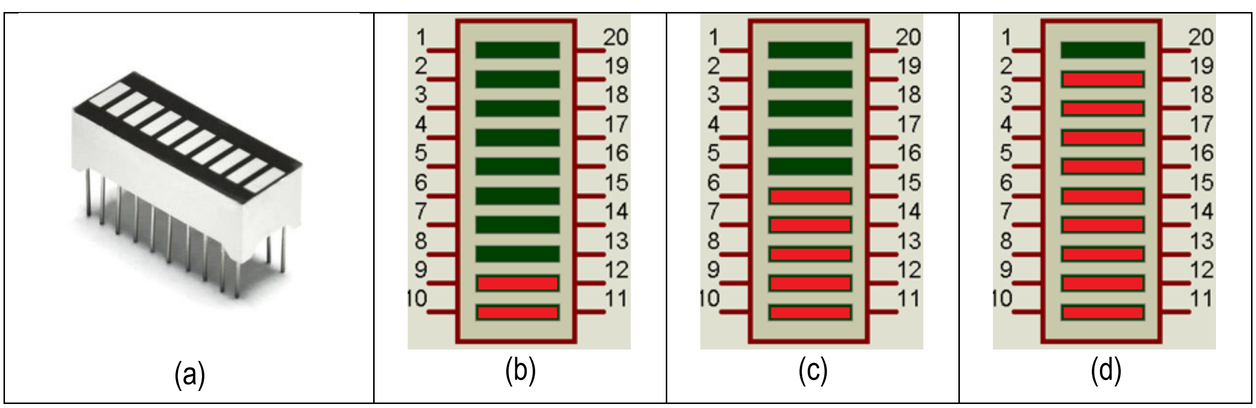

Figure 3 – LED Barograph (a) An actual barograph module (b) Barometer indicating 10mm rainfall (in Proteus Simulator) (c) Barometer indicating 25mm rainfall (d) Barometer indicating 45mm rainfall

Given the data collected by the sensors, we have very simple control rules for the irrigation system summarized in Table 1. Each basin has an automatic valve that will open or close to irrigate the basin or stop the water flow. Based on these simple rules, the valves' state is listed. Since Proteus does not have valves, we will simulate the valves using simple Green LEDs. When the LED is lit, this means the valve is open and the water flows. Otherwise, if it is off, the water is cut off.

Table 1 - Rule-Based Controller Design

Rainfall Level	Soil Humidity sensor	B1 Valve	B2 Valve
≥20mm	*Any*	Closed	Closed
Any	*≥60%*	Closed	Closed
<20mm	*≥40%*	Closed	Open
<20mm	*<40%*	Open	Open

The project should employ all knowledge you learnt in this course either in software or hardware. The project must use the minimum number of port pins as well! The Project must be written in PIC assembly language only.

Resources, Downloads, and Adding Sensors to Proteus

Proteus does not have the mentioned sensors built in. You will need to download them and add them to Proteus library first. Once successfully added, you need to add them and interface them to the PIC microcontroller. You have to add the associated *.hex* files into the sensor. Since we don't have an actual soil and actual rain, we need to simulate these by using potentiometers interfaced to the test pin of each sensor. You can find the sensors, and details on how to add them to Proteus and operate them in these two references:

[1] The Engineering Projects Website, "Soil Moisture Sensor Library for Proteus V2.0", https://www.theengineeringprojects.com/2021/05/soil-moisture-sensor-library-for-proteus-v2-0.html (last accessed Feb. 11th, 2024)

[2] The Engineering Projects Website, "Water Sensor Library for Proteus", https://www.theengineeringprojects.com/2020/07/water-sensor-library-for-proteus.html (last accessed Feb. 11th, 2024)

[3] MPLAB:
https://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_IDE_8_92.zip

[4] Proteus Professional 8.16 SP3:
https://drive.google.com/drive/folders/1NiyITxChJJhNCCy8aXVoGqayXZwMak27?usp=sharing

When you compile your code in MPLAB, in the same project folder, a hex file will be generated. This is the file that is used to program the PIC in real life or in loaded in Proteus. In Proteus, you double click on the PIC, and there you load the hex file, and you are ready to run your code.

Proteus has oscilloscopes/logic analyzers which you can use to view any wave form.

BONUSES
Competition Bonus, and Early Discussion Bonus

The project has three optional bonuses. Students can get all bonuses. **However, in order to be eligible for the bonuses, your code must be fully working for all required specifications, and all parts must be demonstrated as functioning correctly in Proteus**. These bonuses are:

- **Smallest/Fastest Code Competition Bonus (Up to one mark) –** Fully functioning codes are eligible only.
- **Early Discussion Bonus (Up to 1 Mark) -** Fully functioning codes are eligible only.
- **C Project:** In addition to the assembly project, also present a fully functioning code written in Embedded C.

If any student in the project gets less than 3 out of 5 in the discussion part, (s)he will not be entitled to get the bonus!

Smallest/Fastest Code Bonus (Up to one mark)

It is of utmost importance to write codes which are minimum in size and execute quickly. I ask you to use subroutines, modular design and functional reuse whenever possible (functions). In some cases, the use of indirect addressing (FSR and INDF) can be helpful in reducing code size and increasing speed. It is important to not forget to use **functional comments.**

During the discussion, I will ask you to show me on MPLAB the total program size / speed on the PIC. *I will collect this data for all groups with **fully working projects** and assign bonuses as follows:*

Shortest code	1 Mark Bonus		Fastest code	1 Mark Bonus
2nd shortest code	0.5 Marks Bonus		2nd Fastest code	0.5 Marks Bonus

Early Discussion Bonus (Up to one mark)

Project Submission will be midnight Saturday May 25th, 2024. Official discussions will take place on the week of the 26th of May 2024.

- Students who will finish, submit, and **discuss** their project between 19th and 23rd May will get 0.5 Mark bonus.
- Students who will finish, submit, and **discuss** their project before May 19th will get 1 Mark bonus.

Embedded C Project (Up to two marks)

For this bonus, once you finish a working project in assembly language, you can also present a working project in Embedded C. You are already familiar with C++, so working with C should not be difficult. You need to download the Hi-Tech embedded C compiler lite version, and then select it in your project. You can download the C compiler using this link: https://drive.google.com/file/d/1D2_CySfnn2H4jsHmnAuXSw4B32uj2kO3/view?usp=sharing

A quick introduction and review of Embedded C, and building an MPLAB Project using the Hi-Tech compiler can be found here:

https://drive.google.com/file/d/1o2UyYe-tce5g4ydWn6qn_6Kr6Q_w0IKL/view?usp=sharing

How to do Interrupts ISR in C:

https://embetronicx.com/tutorials/microcontrollers/pic16f877a/pic16f877a-interrupt-tutorial/

Plagiarism, Cheating, Generative AI Policy

If any two groups have suspiciously similar codes and exact size, both groups will get a zero and no bonus. Plagiarism is not tolerated and if I suspect any code has been written by someone else, the entire group will get a zero.

Submitting a code written entirely by generative AI tools will immediately result in a grade of zero. Most of the time, generative AI tools will get the idea but implement a code that does not function. You may use them to guide you, but not for immediate submission. My discussion questions will ask you to modify and change the code on the spot, so you need to fully understand the code!

Signup Sheet

The project must be done by **three** students within the allotted time frame. Kindly divide yourself into groups and signup your group names and individual names in the following Google sheet Embedded Course Project Groups

Important Notes

- Start as early as possible on your project, though the project description sounds simple, there is inherent complexity in both hardware and software aspects, so do not underestimate the time it needs, you might have some problems along the way which you will have to resolve!
- Never think of buying a model or commissioning someone to do it for you, not only will you get a zero in the project, but also your act will be considered as a direct violation to JU laws and your actions shall be reported as cheating in the final exam!
- **Code sharing between groups is NOT allowed and leads to 0 points.**
- If you acquire a ***part*** of your software from a book, website, *etc*. Kindly reference it properly, else it will be considered as plagiarism.
- You are only allowed to base your project on PIC16F877a.
- All programming must be done in **PIC ASSEMBLY** language only; using high level languages in the project will get you a Zero.
- Students are not allowed to move between groups once they are formed, so choose your group carefully from the beginning! ***We are not responsible if your colleagues in the group chose to drop the class, we will not allow you to join another group!***

Grading Rubric

Non-Compliance with project requirements: Students who will not adhere to the project software and hardware specifications and requirements will get a zero. Students who do not write in PIC assembly will get a zero.

The project will be graded out of 15, then converted to 10			Grade
1	Proteus (3.5 Marks) • Correct Connections (1 Mark) • Working Model (2.5 Mark)	**No Proteus → Zero** **Schematic** • Incorrect or Incomplete connections (0.5 / 1) • Correct and complete connections (1 / 1) **Working Model (ON/OFF Grading)** • Valves working per specifications - 4 States (2 Marks) • LED barograph working correctly (0.5)	
2	Functional Comments (1 Mark) Your work should be fully documented. *Each subroutine/macro should have its inputs outputs documented; its purpose and algorithm (if any) explained.*	• Comments are written correctly and describe the entire code parts (1) • Comments are written correctly but are not comprehensive (0.5) • Comments are written incorrectly (0)	
3	Coding (5.5 Marks) • Timer and Timer Interrupt (1 Mark) • ADC Setup, Channel Switching, Operation and ADC Interrupt (2 Marks) • Correct Interrupt Setup (1 Mark) • Barograph + 7-Seg. (1 Mark) • Manual Irrigation Mode (0.5 Mark)	Code Correctness for each part.	
4	Student Understanding and Contribution (Individual mark per student) (5 Marks)	**Subject to evaluator** Students must be able to answer any question related to the Proteus Design, Proteus Environment, MPLAB, and the submitted program. Students must demonstrate that they fully understand the entire code, and be able to modify/correct it, or argue on how to extend it if asked to. Students must be able to properly answer the questions and have good communication skills. Students must explain the subroutines, the algorithms, the program flow and how everything is connected to each other. Most importantly, students mark will be largely affected by their contribution. Students who have little contribution will get lower marks. ***Students who fail to pass 3 / 5 will not be eligible to the bonus mark if any is rewarded.***	
5	Bonuses (up to 4 grades)	Early Discussion Bonus (May 19^{th} – 23^{rd}) +0.5 Early Discussion Bonus (Before May 19^{th}) +1 Competition Bonus, +0.5 or +1	