A close up of a logo

Description automatically generated

**Embedded Systems 1 Project**

**Smart Self-Watering Flowerpot**

Lab Completed: 7th May 2020

Submitted by: Dijesh Pradhan

ID:1001516650

**Introduction:**

**Project Overview**

In this project I built a device to monitor the moisture of the soil in a flower pot. In this flower pot we set up a moisture sensor which checks if the soil needs watering. If the soil need watering and the time of the day lies in the watering period then water is pumped out of a reservoir. In this project the water level of the reservoir is also checked and if the water is low, we have set up a speaker with a unique tone which will alert us. We also monitor the light levels and the battery level of the design.

**Project Status**

This project was completed within the given time and was demoed on May 7, 2020. There were no unresolved problems.

**Report Overview**

In this report, I will go through all the details and the parts used to design the Smart Self-Watering Pot. I would go through the brief explanation of the circuits and software I developed.

**Hardware requirements:**

|  |  |
| --- | --- |
| Part | Quantity |
| EK-TM4C123GXL controller board | 1 |
| Pump | 1 |
| Brass rods 1/8” OD x 3” long (electrodes) | 2 |
| Wooden block 1.5” x 0.5” x 0.5” (holds electrodes) | 1 |
| 1’ Copper foil (as capacitors) | 2 |
| 100kohm resistor (capacitive current source, battery  sense) | 2 |
| TEPT5600 ambient light sensor | 1 |
| 1kohm resistor (emitter resistor for light sensor) | 1 |
| 10kohm resistor (for soil sensor) | 1 |
| 100ohm resistor (speaker current limit) | 1 |
| 470ohm resistor (speaker, motor driver base resistor,  deintegrate base resistor) | 3 |
| 10uF capacitor (speaker AC coupler) | 1 |
| 2N3904 (speaker driver, deintegration) | 1 |
| FQP20N06L (logic-level MOSFET for pump motor driver) | 1 |
| 47kohm resistor (MOSFET gate pull-down, battery sense) | 2 |
| 1N4004 (motor flyback diode) | 1 |
| Various colors of wire | 1 |
| 4 D cell battery holder | 1 |

**Project Details**

**Phase 1:**

In the first phase of the project, a terminal interface was built so that the design would take commands and process the commands. For the terminal interface, I used UART was my platform. UART stands for Universal Asynchronous Receiver/Transmitter. The main purpose of UART is to receive and transmit serial data. I initialized the UART registers using the data sheet provided for the arm controller (EK-TM4C123GXL). In this phase the characters were first separated in three different types- alpha, numeric and delimiter. The letter before the first character was considered to be a delimiter. After the separation of these three types, the delimeters were changed to a NULL so that it would be easier for us to separate out commands like Pump ON. Two functions char\* getFieldString(USER\_DATA\* data, uint8\_t fieldNumber) and int32\_t getFieldInteger(USER\_DATA\* data, uint8\_t fieldNumber) which would return a String and Integer from the fieldNumber and if the field number is not in range it would return NULL. I also added a function isCommand which would check if the first word matches the command with the minimum arguments required for the command like Pump ON has 1 argument after the Pump and if only Pump is hit without any argument it would not work.

**Phase 2:**

In this phase of the project, I added the ability to measure the amount of water in a container using the circuit given in class. I initialized the hardware so that it can support the comparator 0 with an internal reference of 2.469V and an external input, C0-, which is connected to the sensor capacitor as shown in the figure. I configured DEINT so that I could control it’s pin on a GPIO. A timer was set so that it would help to measure the count/time which would help me to calculate the volume. In the software to calculate the volume using timer I pulled the DEINT high for 1000 microseconds, using a function waitMicrosecond(). After the DEINT was pulled high which deintegrated the capacitor, I set the DEINT low which would start the capacitor to charge and the timer was started immediately. As soon as I turned on the timer, I measured the time requied to charge the capacitor and as the value is proportional to the capacitance and the volume. I measured the time when the reservoir was empty and when it was full and derived this equation:

volume= (0.5330\*(timer-322))

The timer in the above equation is the time which the capacitor took to charge up. In this way, the volume of the reservoir was measured.

**Phaser 3:**

In this phase I was able to add the ability to measure the intensity of light and the moisture of the soil in a container using the circuit given in class. To support the use of the sensors, first we need to initialize the ADC which means Analog to Digital Convertor. This was used so that the analog values that we get from the sensors could be changed to a digital one. For the sensors I initialized the hardware so that it could support ADC0 to make measurements for the light sensor, moisture sensor and the battery voltage measurement. For the light sensor, AIN2 was used as the analog input for the light sensor. The analog input was initialized and configured. For this part of the project ADC0Ss3 was configured so that we could take few samples of the voltage reading and find the voltage reading from the light sensor and moisture sensor. TEPT5600 ambient light sensor was used as the sensor for the light. For the light sensor, the voltage when there is light was found to be 3.2V and the voltage when there is no light was found to be 0V. Same goes for the moisture sensor, when the water was full in the brass rods it was found to be 3.2V and 0V when it was dry. For the moisture sensor AIN1 was used in the software as the analog input. The following equation was used to convert the samples collected which would give out the voltage:

Voltage = (((raw+0.5) / 4096.0 )\*3.3) ; where raw is the samples and voltage are for the respective sensors.

For the battery sensor everything was the same except the change of the analog input which was AIN0 in my software.

**Phase 4:**

In this part of the lab, I added the ability to run the pump to give out water from the reservoir. I also added the ability to play alerts on the speaker using the circuit given in class. The hardware was initialized in the program to support the digital outputs on the pump and the speaker. For the pump two functions were written, enablePump() and disablePump(). In the enablePump() function I set the MOTOR which is the bitband of the GPIO to high (MOTOR=1) which would turn on the PUMP and in the function disablePump(), the MOTOR was set low (MOTOR=0) which would turn off the motor. The UART interface was built to support the command Pump ON and Pump OFF.

I set a timer2 for this phase of the project which would be a crucial part for the speaker to make my own alert. An interrupt was created in the css.startup file so that the timer would give interrupt to create the sound. Timer1ISR function is used to toggle the speaker GPIO pin. In the playWaterLowAlert() function and playBatteryLowAlert() we started the timer and load a value in the timer to play the sound and waited for some time and the turned off the timer. This process can be repeated as many times as we would like to create the tones.

**Phase 5:**

In this phase, we initialized the hibernation module RTC for operation so that we can know the real time of a day. In this we initialized the hibernation module and load the value in the HIB which is the seconds that are converted from the start of the program. This would increase in every second and which would tell us the exact time we are in right now. This was built so that the Pump would work only in a given time frame and the pump would not start in a random time like middle of the night. We set the H and M as the time when we start the program and H1 M1 to be from 9 am to H2 M2 to be at 5 pm. If the H M lies in between H1 M1 and H2 M2 and the moisture of the soil is less than the Pump would be enabled. In Phase 1, we used a blocking function in the program which would only work if anything was pressed. But in this stage, we improvised and made the program to work like checking the sensors even when no one has given a command to do so.

**Phase 6:**

In this extra credit part, I included a Command History which would store the History of the moisture sensor, light sensor and volume every time we hit the command status. At first the EEPROM was initialized and configured. As there are 32 blocks and each block has 16 offset where each offset has 32 bit space to store a word which means we can store 512 words in it. I used only one block to store the data in the EEPROM where moisture, light and volume were stored simultaneously.

**Conclusion:**

This project was a great learning experience which helped me understand about embedded systems more deeply. The project took about 8-9 weeks to complete fully. Even in this difficult time of pandemic I was able to finish the project with the help of Dr. Jason Losh and Fahad. This really motivated to take more of computer engineering classes in the future.