GARBAGE MONITORING SYSTEM USING GSM

Rohith Sali

Srishti Khare

Final Project Report

ECEN 5613 Embedded System Design

December 10, 2016

TABLE OF CONTENTS

1. INTRODUCTION

Overflowing garbage bins have been another cause of concern for residents in developing countries. With increase in population, the scenario of cleanliness with respect to garbage management is degrading tremendously. With the already prevailing diseases, the open containers are proving to be a breeding place for germs. Traditionally, municipalities operate on weekly routes to pick up trash and recyclables on designated days, regardless of whether the containers are full or not.

Figure 1: Traditional way of Garbage Collection

Our project aims to optimize waste collection and ultimately reduce fuel consumption and time. Basically, it will shoot sonar waves to know how much stuff is inside the container. Data collected from the sensors are sent to the microcontroller and displayed on the LCD. Microcontroller is connected to GSM module. Once the bin is full GSM module sends out SMS indicating which bin is filled over a cellular network indicating it to the user.

Essentially, this project is about collecting the most amount of materials in the least amount of time to reduce costs and emissions along the way. Furthermore, this project is supposed to work with any type of container and any type of waste, including mixed materials, paper, glass, metals. Thus, there will be saving in fossil fuel due to optimized route for collecting garbage and also thus transportation cost.

Figure 2: Smart way of Garbage Collection

1.1 SYSTEM OVERVIEW

Since there are lot of problems regarding garbage, and the unhygienic conditions occur due to overflowing of the garbage, we decided to come with a simple solution to avoid such conditions as final project.
This design incorporates several new components spanning hardware and software. Fig1.1: Shows the block diagram of the system. Ultrasonic sensors are connected to PSoc 5lp. The processor acts as the CPU of the system which consists of Arm Cortex m3 processor. It processes the time duration obtained from the sensors to distance and the distance is displayed on the LCD. Processor prompts the GSM module to send a SMS to the user when certain distance value is reached.

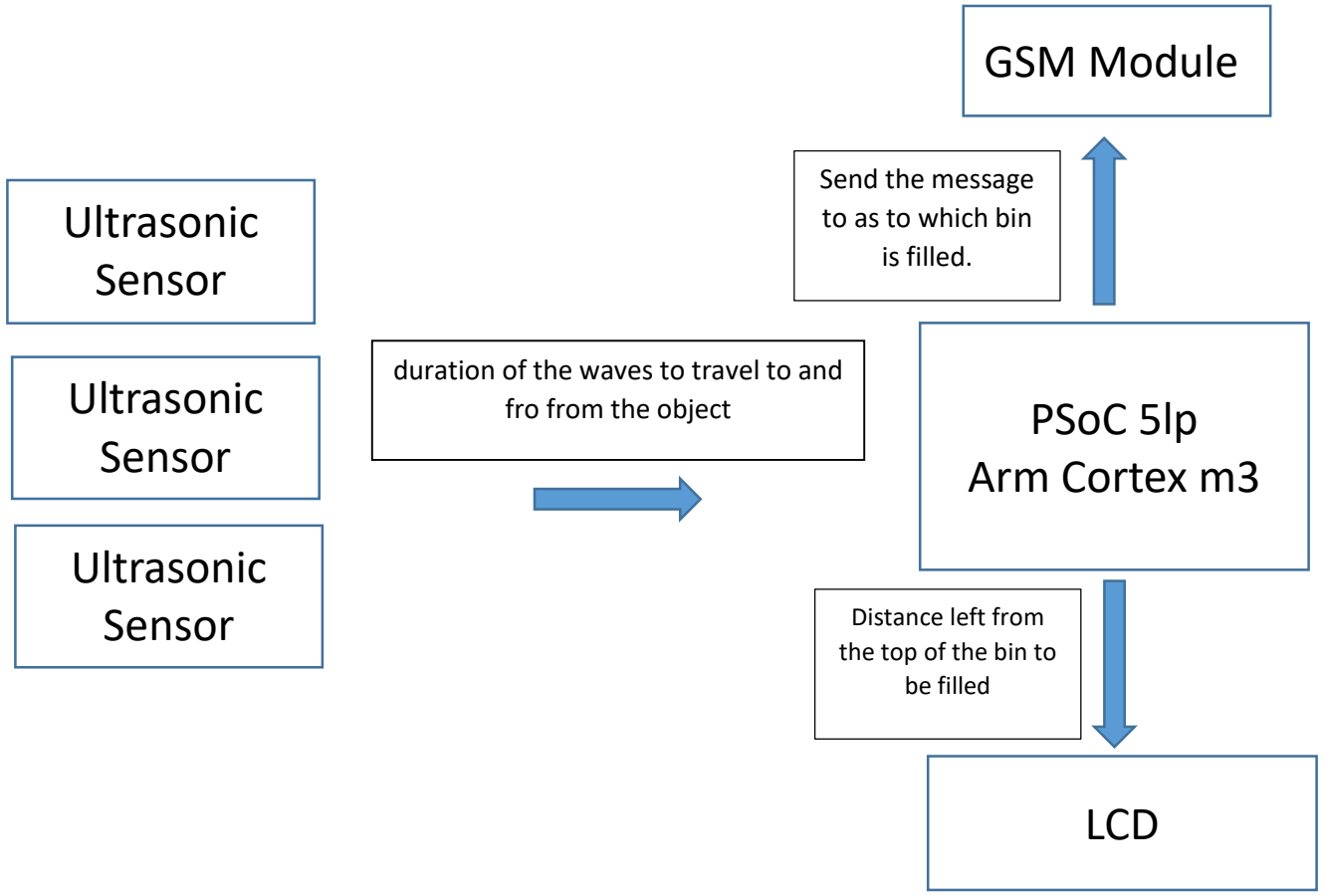


Figure 3: Project overview, functional block diagram

2. HARDWARE

2.1 PSOC 5LP

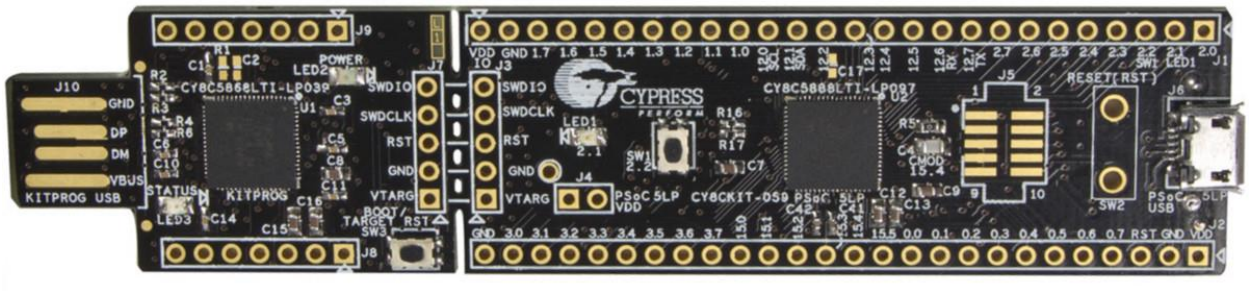


Figure 4: PSoC 5LP Development Board

We wanted to work on hardware that we had very little familiarity with and which has an ARM processor. Hence we decided to work on development this board with ARM Cortex M3 processor, using PSoC Creator IDE.

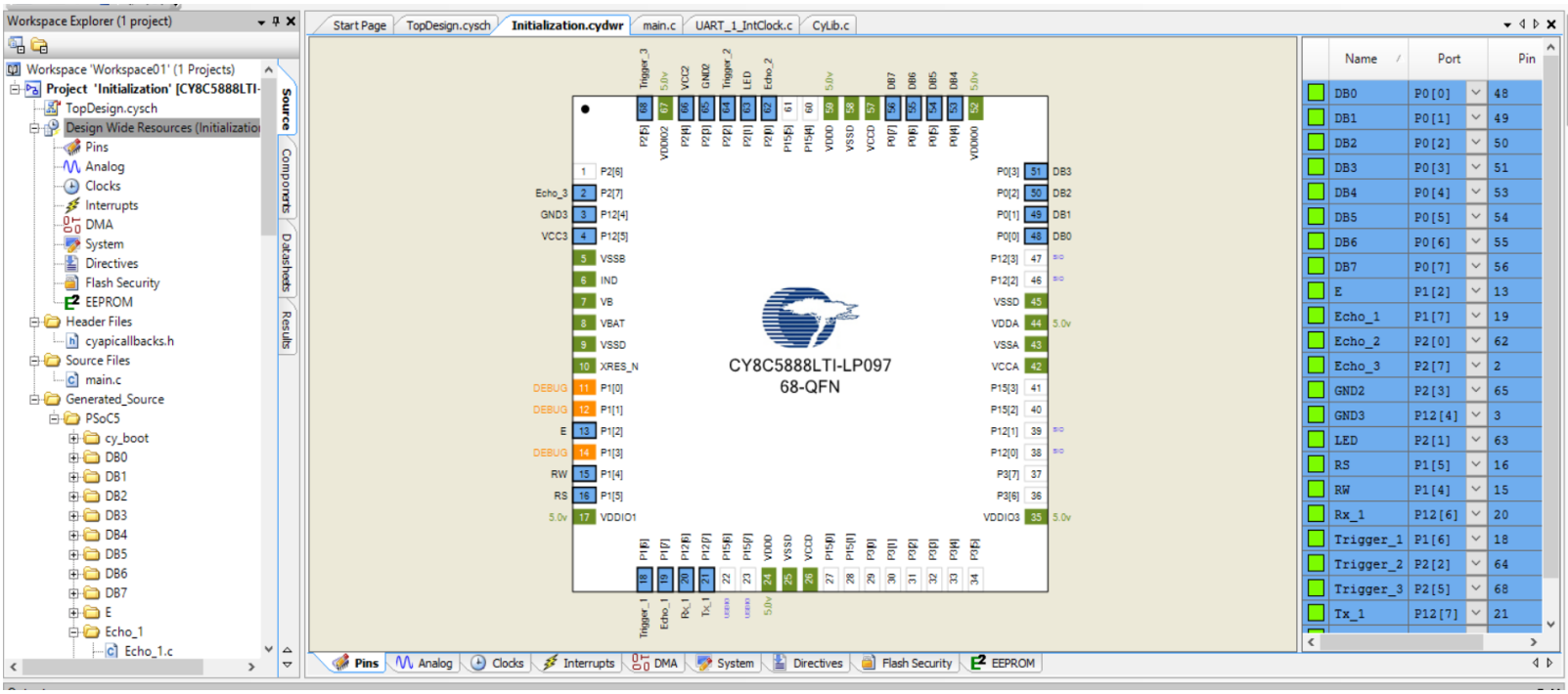


Figure 5: PSoC Creator 4 device pin assignment

Though there are several modules available in PSoC creator to add to be the schematic and configure as per requirement, we bit-banged the functionality of timers and PWM, and learned immensely through the process.

2.2 HC SR-04 ULTRASONIC SENSOR

Figure 6: HC SR04 Ultrasonic Sensor

The HC-SR04 ultrasonic sensor uses sonar to determine distance. It offers excellent non-contact range detection with high accuracy and stable readings. It can measure from 2cm to 400 cm or 1" to 13 feet. It operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect).

Features:

Power Supply: +5V DC

Quiescent Current: <2mA

Working Current: 15mA

Effectual Angle: <15°

Ranging Distance: 2cm – 400 cm/1" - 13ft

Resolution: 0.3 cm

Measuring Angle: 30 degree

Trigger Input Pulse width: 10uS

2.2.1 PRODUCT SPECIFICATION AND LIMITATIONS

Parameter	Min	Typ.	Max	Unit
Operating Voltage	4.50	5.0	5.5	V
Quiescent Current	1.5	2	2.5	mA
Working Current	10	15	20	mA
Ultrasonic Frequency	-	40	-	kHz

Table 1: HC SR04 Product Specifications and Limitations

2.2.2 OPERATION

The timing diagram of HC-SR04 is shown. To start measurement, Trig of SR04 must receive a pulse of high (5V) for at least 10us, this will initiate the sensor will transmit out 8 cycle of ultrasonic burst at 40kHz and wait for the reflected ultrasonic burst. When the sensor detected ultrasonic from receiver, it will set the Echo pin to high (5V) and delay for a period (width) which proportion to distance. To obtain the distance, measure the width (Ton) of Echo pin.

Time=Time/1000000; //convert Time to seconds by dividing by 1000000 (microseconds in a second)

Time=Time/3600; //convert time to hours by dividing by 3600 (seconds in an hour)

Distance= Speed of sound * Time; //This will be in miles, since speed of sound (776.5) was miles per hour

Distance=Distance/2; //Waves travels to target and back from target, so you must divide by 2 for actual target distance.

Distance= Distance*63360; //Convert miles to inches by multiplying by 63360 (inches per mile)

2.2.3 TIMING DIAGRAM

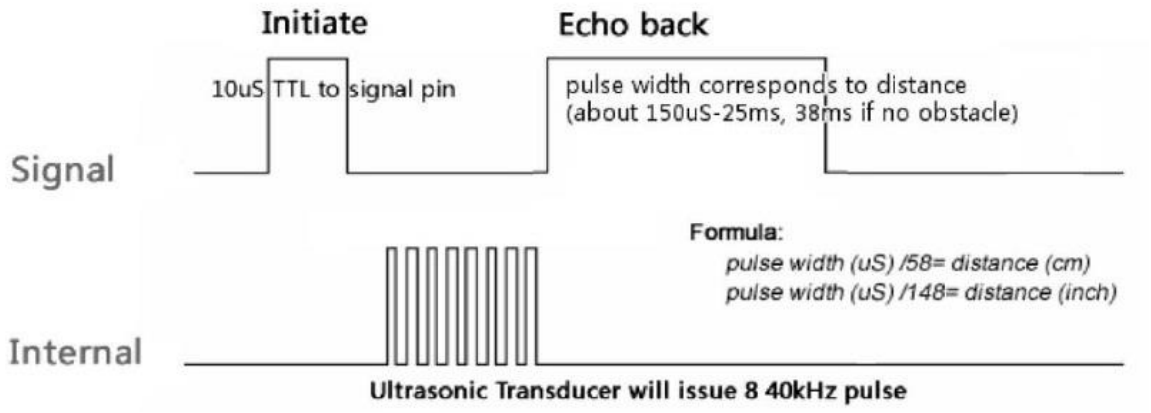


Figure 7: HC SR04 Timing Diagram

2.2.4 PINS CONNECTIONS

HC SR-04	PSOC 5LP
VCC	5V supply pin and GPIO pins set HIGH
GND	GND pins
Echo (for each sensor)	GPIO pins P1[7], P2[0], P2[7]
Trigger (for each sensor)	GPIO pins P1[6], P2[2], P2[5]

Table 2: HC SR04 Pin Connections with PSoC 5LP Dev Board

NOTE →

We observed that connecting Trigger pin directly to 5V supply on board and not to a PWM (or equivalent code to generate a pulse wave) causes a similar triggering mechanism, though we have written a c code with appropriate delays to trigger the sensor.

2.2.5 CODE FOR ULTRASONIC SENSOR DISTANCE CALCULATION:

```
unsigned long pulseIn_1(void) { //fuction to calculate the Echo pin HIGH
duration

    unsigned long pulseWidth_1 = 0; //to count the duration for which Echo
stays high
    unsigned long loopCount_1 = 0;  //loop counter
    unsigned long loopMax_1 = 5000000;  //loopcounter upper limit

    // While the Echo pin is not HIGH, make sure the timeout hasn't been
reached
    while ((Echo_1_Read()) != 1) {
        LED_Write(0);
        if (loopCount_1++ == loopMax_1) {
            return 0;
        }
    }

    // When the Echo pin is HIGH, increment the counter while still keeping
track of the timeout
    while ((Echo_1_Read()) == 1) {
        LED_Write(1);   //make LED high on board when Echo goes high      if
(loopCount_1++ == loopMax_1) {
            return 0;
        }
        pulseWidth_1++;
    }

    // Return the pulse time in microseconds
    return pulseWidth_1 * 2.36; // Calculated the pulseWidth++ loop to be
about 2.36uS in length
```

We took the inspiration to code this function form the in-built Arduino function.

2.2.6 CODE FOR ULTRASONIC SENSOR TRIGGER PIN:

```
Trigger_1_Write(0); //code to trigger the Echo pin (applied to Trigger pin)
        CyDelayUs(2);
        Trigger_1_Write(1);
        CyDelayUs(10);
        Trigger_1_Write(0);
        CyDelayUs(2);
```

2.2.7 CODE FOR ULTRASONIC SENSOR OUTPUT IN INCHES:

```
duration_1 = pulseIn_1();   //Reading microseconds value during which Echo
pin stays HIGH

duration_1 = duration_1*0.017;  //Conversion of microseconds to inches
```

2.3 GSM MODULE ADAFRUIT FONA 808

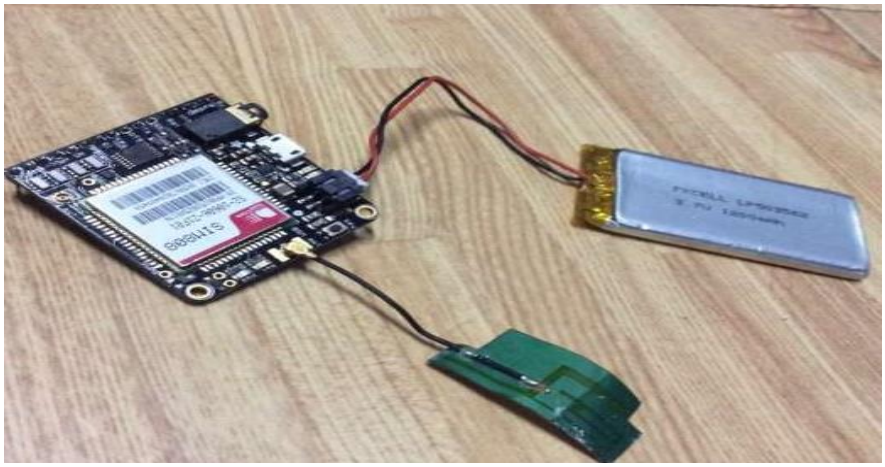


Figure 8: GSM Module

Adafruit fona 808 is a module which contains GPS, GSM which lets us add SMS, GPS and data.

It consists of an onboard LiPoly battery charging circuitry which can be charged with 500mA charger.

By using Putty, we can test the AT commands. This module echoes the characters back. We have initialized the auto Baud by sending "**AT+return**".
ATI - Get the module name and revision.
AT+CMEE=2 - Turn on verbose errors (handy for trying out commands!)
AT+CCID - get the SIM card number - this tests that the SIM card is found OK and we verified the number written on the card.

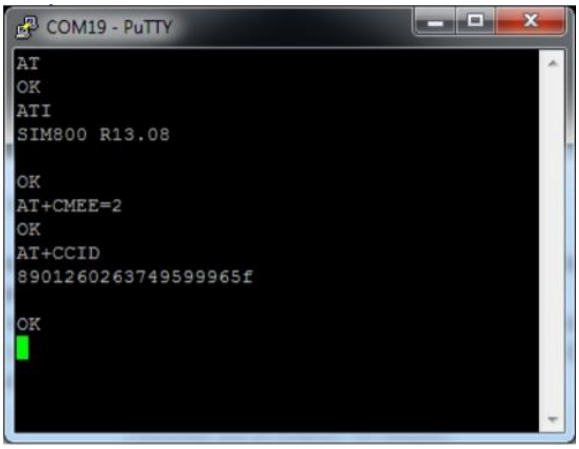


Figure 9: GSM Commands

2.3.1 TEST SETUP

Some tests to verify the setup:

AT+COPS: Check that you're connected to the network, in this case T-Mobile

AT+CSQ: Check the 'signal strength' - the first # is dB strength, we obtained 14, it should be higher than or 5. Higher is better. Of course it depends on antenna and location!

AT+CBC - will return the lipo battery state. The second number is the % full (in this case its 92%) and the third number is the actual voltage in mV.

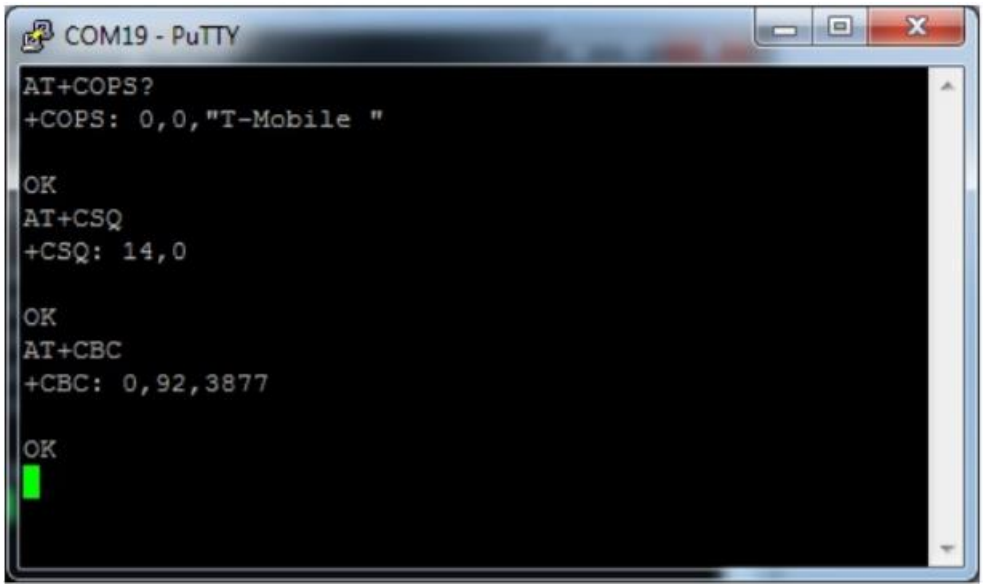


Figure 10: GSM Commands

2.3.2 TEST SETUP SEND SMS

AT+CMGF=1 <-- this will set it to TEXT mode not PDU (data) mode.

AT+CMGS="nnnnnn" <-- send a text message! You will get a '>' prompt for typing.

After typing the message and when it is done send a [Control-Z] on an empty line to send the SMS.

2.3.3 PIN CONNECTIONS

GSM Module	PSoC 5LP
GND	GND pin
VCC 3.3V	USB charging
TX	RX pin of the board, P12[7]
RX	TX pin of the board, P12[6]

Table 3: GSM Module Pin Connections

2.3.4 CODE FOR GSM MESSAGE TRANSMISSION:

```
UART_1_PutString("AT");//sent to access/init GSM module
        CyDelay(500);//500 ms delay
        UART_1_PutChar(0x0D);//carriage return
        CyDelay(500);//500 ms delay
        UART_1_PutString("AT+CMGF=1");//sent to set GSM to TEXT mode not PDU
(data) mode.
        CyDelay(500);//500 ms delay
        UART_1_PutChar(0x0D);//carriage return
        CyDelay(500);
        UART_1_PutString("AT+CMGS=\"+17203279021\"");//sent to send a text
message! You will get a '>' prompt for typing.
        CyDelay(500); //500 ms delay
        UART_1_PutChar(0x0D);//carriage return
        CyDelay(500);
        UART_1_PutString("Trash Can 3: Completely filled!");     //message
sent to cellphone
        CyDelay(100); //100 ms delay
        UART_1_PutChar(0x1A);//ctrl Z
        CyDelay(500); //500 ms delay
```

2.3.5 USE OF UART MODULE

A UART is used in the schematic of PSoC Creator IDE to perform GSM operations and send commands.

UART is configured and used to send GSM commands.

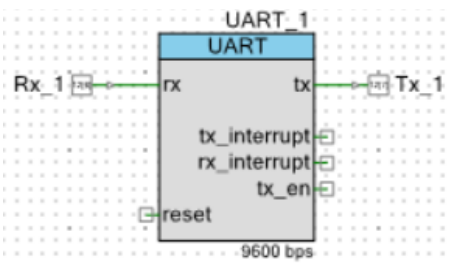


Figure 11: UART Module used in schematic of PSoC Creator IDE

2.4 HITACHI HD44780U LCD CONTROLLER

We used the LCD controller which was given to us as a part of the ESD Lab Kit and used by us in previous labs.

Figure 12: 16x4 LCD Controller

We interfaced the 16x4 LCD to display the readings of each of the three trash bins in inches and change in real time as per the operations performed.

All of the Port 0 pins were assigned to the data pins of the LCD controller.

2.4.1 PIN CONNECTIONS

HD44780U	PSOC 5LP
VSS	GND
VDD	5V supply pin
VEE	GND
RS	P1[5]
RW	P1[4]
E	P1[2]
DB0	P0[0]
DB1	P0[1]
DB2	P0[2]
DB3	P0[3]
DB4	P0[4]
DB5	P0[5]
DB6	P0[6]
DB7	P0[7]

Table 4: 16x4 LCD Controller Pin Connections

3. ERROR ANALYSIS

3.1 ESP 8266

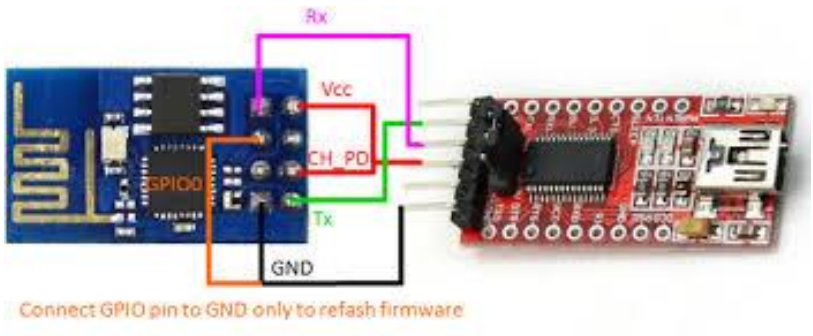
We wanted to interface ESP8266 Wi-Fi module to monitor the system in real time.

The ESP8266 is a low-cost Wi-Fi chip with full TCP/IP stack and MCU (Micro Controller Unit) capability.

Features

- 1MB Flash Memory
- Integrated low power 32-bit CPU could be used as application processor
- SDIO 1.1 / 2.0, SPI, UART
- STBC, 1×1 MIMO, 2×1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)

We interfaced the module with USB to Serial IC FTDI board which gives 3.3V. We tried to flash the module with simple program using Arduino IDE to read data from a website. We need to access the bootloader to upload the program.



After we flash the firmware we disconnect the GPIO pin from Ground. The module runs at 57600 baud when it is in bootloader. After flashing the firmware, the watchdog timer was getting reset and it was still in Bootloader and was not coming out of it. After doing a bit of research low current is one of the causes for the watchdog to reset. We tried connecting it to external power source but result remained the same. We flashed the bootloader and again the result was same. We finally contacted sparkfun and they replied saying they currently don't have modules to exchange and there is a problem with the module.

3.2 PSoC CREATOR TIMER MODULE

We initially used the Timer module in the schematic of PSoC Creator IDE. Since the ultrasonic sensor works on the principle that its Echo pin stays HIGH whilst it the receiver is waiting for the reflected ultrasonic waves, we used a Timer to calculate that duration. Though the timer started on time (as the Echo pin went high), it did not stop as the pin went low in value. This caused constantly changing values of the readings of the senor on the LCD display.

We tried a lot of combinations of the timer start\stop command placements and even switched to counters, but still the readings captured were vague and erratic.

Finally, we read about the “PulseIn()” function commonly used in all programs with HC SR04 interfaced with an Arduino board.

We read how the PulseIn() function increments a counter while the Echo pin is high, using the correct data type.

We then bit-banged the function and used that to obtain correct values from the sensor.

Through this experience we learned not only how to use timers and counters for such applications but also how a simple fundamental program does the work when you thought it was going to be really complicated.

4. DEMO SETUP

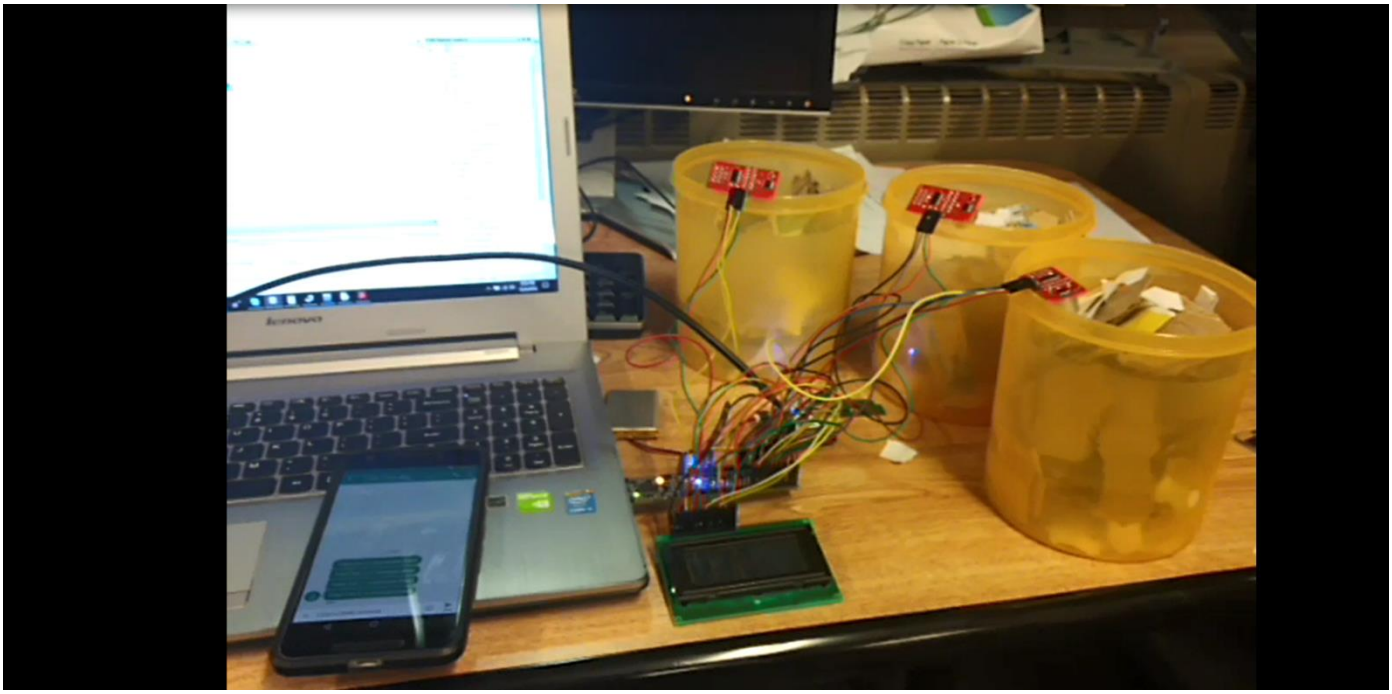
For demo purposes, we used three plastic containers as trash bins, and added bits and pieces of paper & cardboard as trash.

This setup was then connected to a GSM module which would send messages to a cell phone once the respective bins are full (we set threshold values).

We demonstrated how as we put in the cardboard pieces into the containers, the values of the distance of the three sensors from the bins change in real time.

As the bins get full, we receive messages on the cell phone stating the same for each bin respectively.

Here is what the demo setup looked like →



We will be uploading the video of the setup and demo on D2L.

5. CONCLUSION

The entire process of making this project, from the idea to design, coding and then demo setup, was a wonderful and teaching experience to say the least.

We chose this project since we had not worked on this board before and wanted to learn something new. The project looked doable and interesting. We learned both ARM processor coding as well as GSM techniques while working on this project.

Another reason we chose this project getting to interface three ultrasonic sensors, an LCD controller and a GSM module with the ARM dev board. We learned a lot about simple things such as delays in the code for proper calibration and functioning of the three sensors along with the LCD, bit-banging the timer module functionality, to making it all presentable for the demo.

Meeting with issues such as the dysfunction of Wi-Fi module and timer module in PSoC Creator, we learned how to manage our time better, be prepared for the worst and not take anything that looks easy for granted.

After all, filling up containers with bits and pieces of paper, waiting for it to fill up and send you messages is fun, no matter how many things went wrong and how we feel we could have done a much better job.

6. FUTURE SCOPE

Like everything however good, this project of ours too has a lot of scope for improvement.

The placement of the sensor on the trash bin could be one for starters. In our demo, we placed the ultrasonic sensors on the rim of the containers, hence leading to a “blind spot” at the base right below it. This can be avoided by placing the sensor at the center of the lid covering it. This way the sensor will have complete coverage of the area of the bin and will give accurate readings.

The garbage monitoring system can interface aWi-Fi module with ultrasonic sensor to monitor the garbage level readings in real time on the internet. Websites such as IoT Gecko offer UIs for this purpose.

There could also be cases where a light box is placed at the top of the trash bin when the bottom is still empty. This will lead to false readings. This can be taken care of by placing pressure sensors at the base of the bin.

7. REFERENCES

[1] Serial-to-WiFi Tutorial using ESP8266
http://fab.cba.mit.edu/classes/863.14/tutorials/Programming/serialwifi.html

[2] IoT Based Waste Management System - Smart Garbage Monitoring System (City of San Leandro)
us-ignite.org/globalcityteams/actioncluster/fS2k3yUuyxaAZSYjgcEn4Y/

[3] ESP8266 Wi fi module explain and connection by Matej Blagšič in wireless
http://www.instructables.com/id/ESP8266-Wi-fi-module-explain-and-connection/

[4] ESP8266 - Easiest way to program so far (Using Arduino IDE)
http://www.whatimade.today/esp8266-easiest-way-to-program-so-far/

[5] Adafruit FONA 808 Cellular + GSM Breakout
https://learn.adafruit.com/adafruit-fona-808-cellular-plus-gps-breakout/wiring-to-usb

[6] http://www.toptechboy.com/

[7] Microcontroller projects IoT garbage monitoring
http://circuitdigest.com/microcontroller-projects/iot-garbage-monitoring using-arduino-esp8266