

**MKSSS’s**

**Cummins College of Engineering for Women, Pune**

**(An Autonomous Institute Affiliated to Savitribai Phule Pune University)**

**DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING**

OPEN ENDED ASSIGNMENT - REPORT

ON

**FUEL MONITORING SYSTEM**

### SUBMITTED BY

**Piyusha Shendage ( Roll No: 3243 & C. No.:** C 22020111245 **)**

**Vaishnavi Shirsalkar ( Roll No: 3244 & C. No.:** C 22020111246 **)**

**Shivani Patil ( Roll No: 3246 & C. No.:** C 22020111248 **)**

**Tanvi Tambe ( Roll No: 3254 & C. No.:** C 22020111256 **)**

**Shivani Thorve ( Roll No: 3258 & C. No.:** C 22020111260 **)**

**Under the Guidance of :-**

**Prof. Nitin Palan**

**Department of Electronics and Communication Engineering**

**MKSSS Cummins College Of Engineering For Women, Pune.**

**TABLE OF CONTENTS**

**ABSTRACT**

**1. INTRODUCTION**

**2. BLOCK DIAGRAM**

**3. INTERFACING DIAGRAM OF OVERALL SYSTEM**

**4. COMPONENT INFORMATION**

**5. ALGORITHM / FLOWCHART**

**6. SOFTWARE CODING**

# 7. **OUTPUT**

# 8. APPLICATION & ADVANTAGES

# 10. **CONCLUSION**

# 

**ABSTRACT**

To help mitigate the risk of fuel theft and siphoning the proposed fuel monitoring system helps us to calculate the volume of fuel filled into the tank by using the ultrasonic sensor to find the difference between the distance in the initial and final levels of fuel in the tank and then calibrating the value into the actual volume of fuel poured into the tank.

The ultrasonic sensor can give precise readings up to 2 cm to 4m and hence can be used in detecting the level of fuel with an appropriate placement into the tank. This can be considered as an inexpensive and efficient solution to get information related to volume of fuel filled.

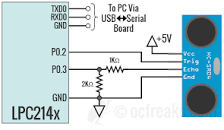
**INTRODUCTION**

Everything in the world is digital, interacting with real-time systems and living a simpler life are both made possible. Automobiles are no longer considered a luxury in today's fast-paced society; instead, they are merely a necessity for the majority of people. This change coincides with the increased demand for fuel, garages, service stations, etc., which creates new opportunities for fraud. The scam at the gas pump is the most prevalent in India.

Normally when we fill fuel in a tank we are not sure about whether we are getting exactly what we have asked for i.e. 1L, 2L etc. This problem is accompanied by modern automobile systems' digital fuel gauges having the drawback of showing a bar or deflecting needle as opposed to the actual amount of gasoline in the rider's fuel tank. Fuel thefts stem from this. Digital fuel meters and anti-theft systems should be installed in vehicles to prevent such issues.

We want to improve user experience by addressing such important issues in our project. The ultrasonic sensor in our model, which we further calibrated to determine the actual amount of fuel being filled in the user's fuel tank, measures the distance between the initial and final fuel levels. This sensor, which incorporates an lpc2148 and a serial port to display the amount of fuel filled on the user interface, simplifies the process of allowing users to determine which stations are scamming them and which are not.

**BLOCK DIAGRAM**

****

In our project we have interface LPC2148 with Ultrasonic ranging module HC-SR04 that provides 2cm - 400cm non-contact measurement function, and has ranging accuracy that can reach to 3mm. Here the trigger pin of ultrasonic is at P0.2 and echo pin at P0.3. We have used a serial port on the user interface part where our fuel in liters is displayed.

**CONNECTIONS**

1. TxD → P0.0
2. RxD → P0.1
3. Trigger Pin → P0.2
4. Echo Pin → P0.3
5. Vcc → 5V

**CALIBRATION**

For calibration of distance and relation of distance increased with respect to liters of fuel filled, we performed a case study of an ACTIVA fuel tank.

Following are the measurements of the fuel tank:

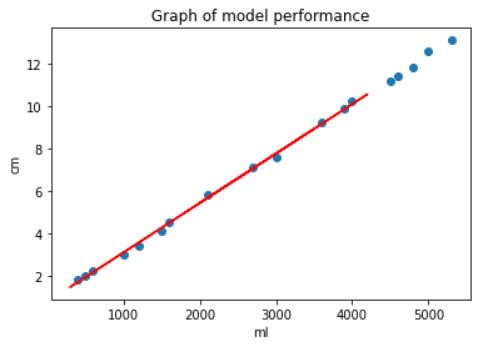
|  |  |
| --- | --- |

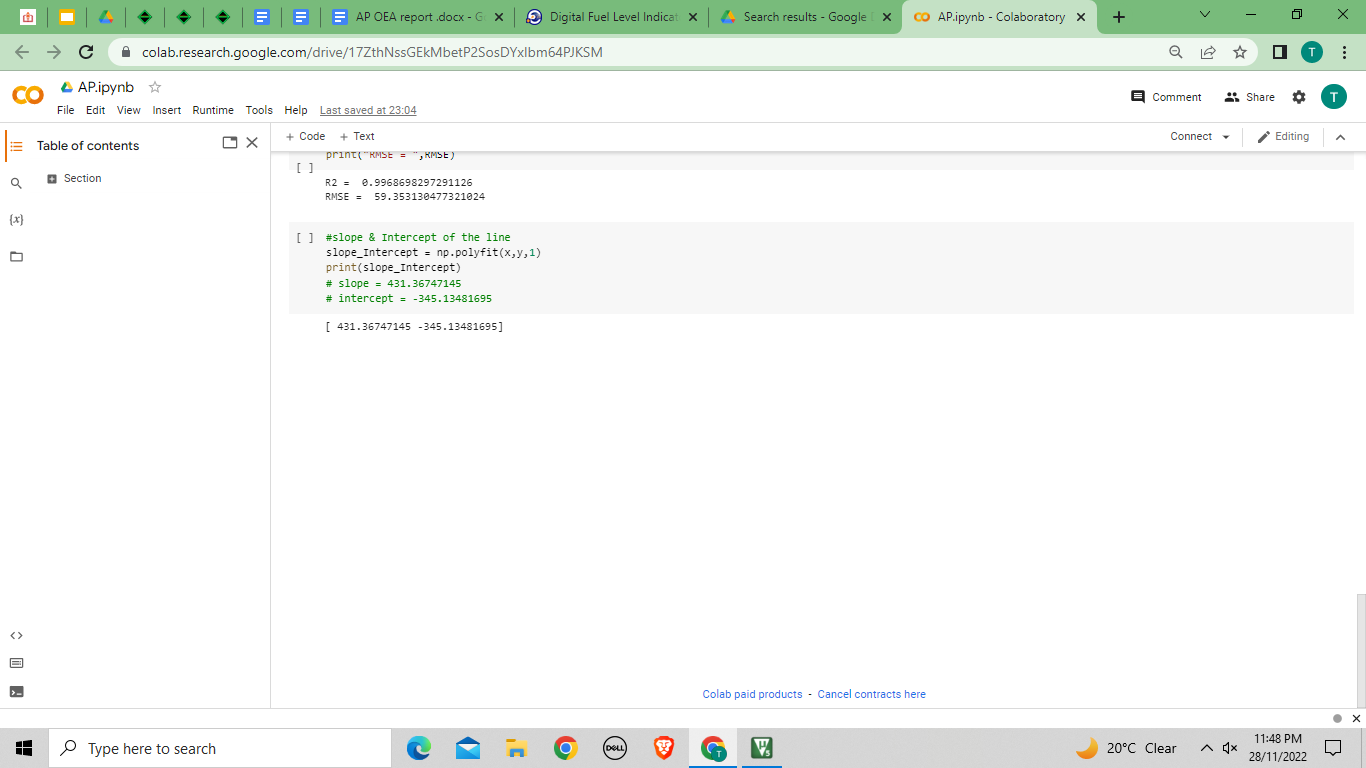
|  |  |
| --- | --- |
|  |  |

We observed the distance filled for each 100 mL increment at every stage and plotted a graph to see the relation between volume and distance.

|  |  |
| --- | --- |

From the graph it can be seen that the relation between volume and distance is linear and we fitted a linear regression model on this dataset to obtain our final volume filled equation.

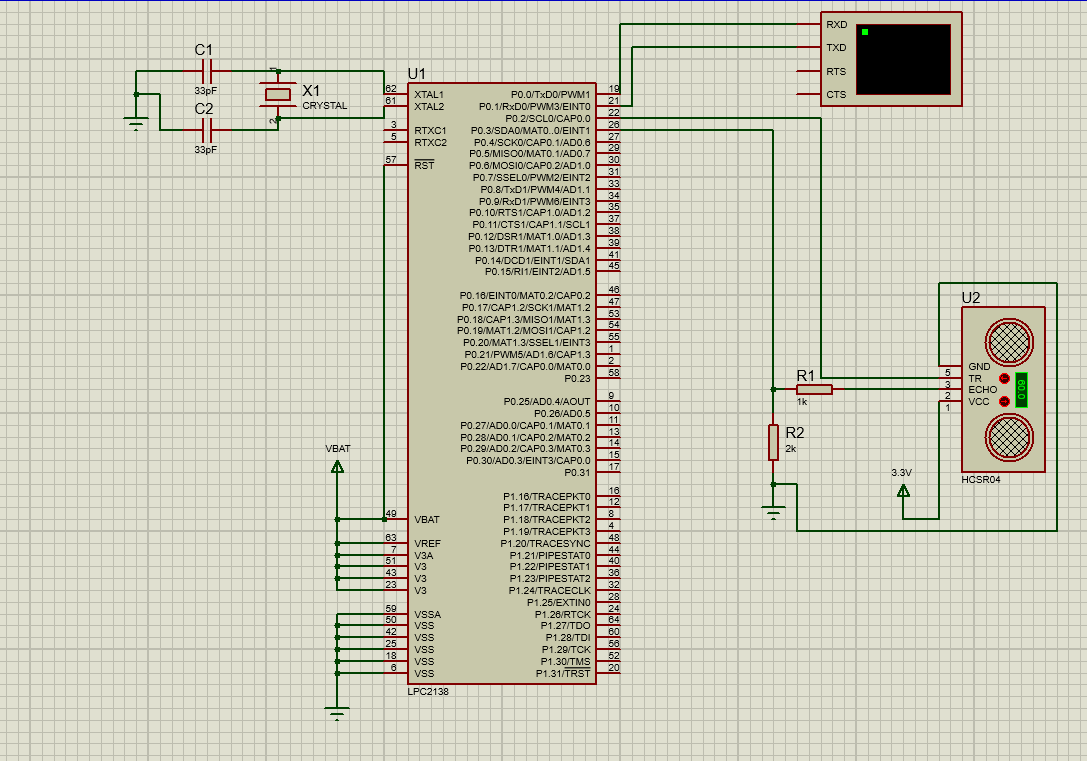




After fitting the model we find our data can be efficiently represented by the **Y = MX + C** equation where M (slope) is 431.36747145 and C (intercept) is -345.13481695.

**Y = ((431.36747145)\*(DISTANCE) + (-345.13481695)**

**INTERFACING DIAGRAM OF OVERALL SYSTEM**

****

**COMPONENT INFORMATION**

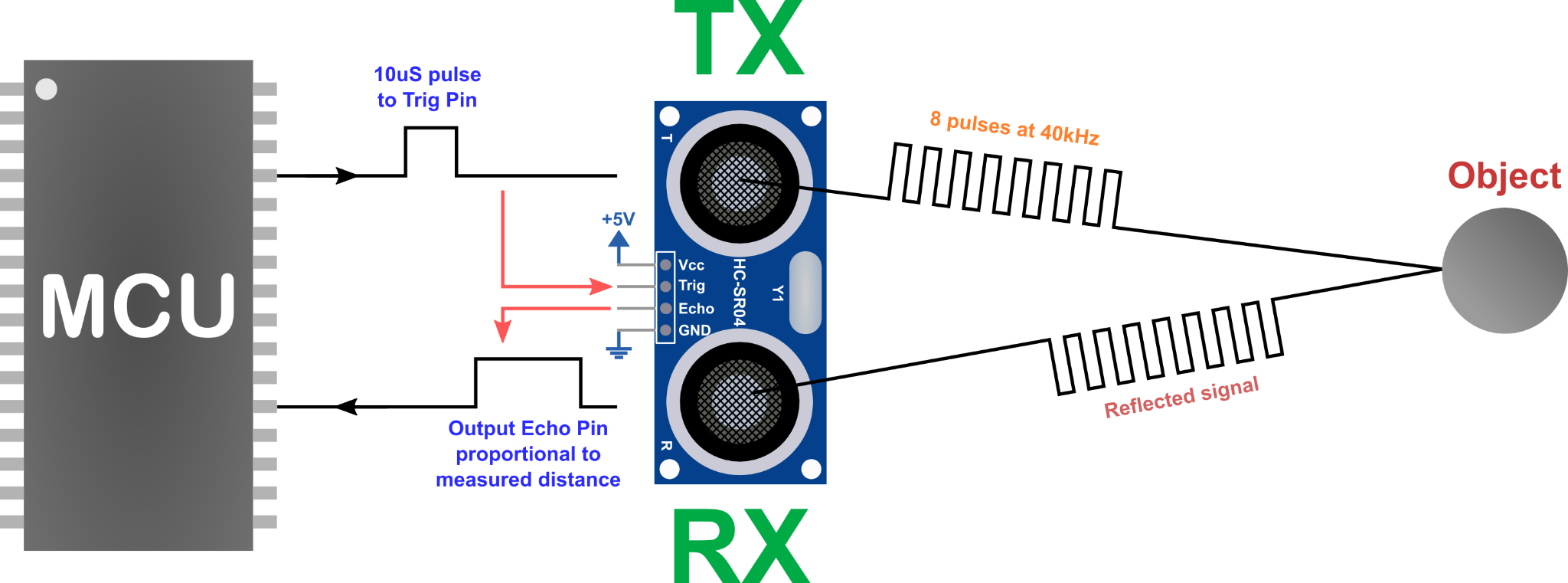
1. **LPC2148**

LPC2148 has In-System Programming/In-Application Programming (ISP/IAP) via on-chip boot loader software, 32kB on chip SRAM and 512kB on chip FLASH memory. This chip has built-in support for up to 2kB USB RAM.

1. **Ultrasonic sensor**

It is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has traveled to and from the target).

In order to calculate the distance between the sensor and the object, the sensor measures the time it takes between the emission of the sound by the transmitter to its contact with the receiver. The formula for this calculation is [D = ½ T x C](https://www.arrow.com/en/research-and-events/articles/ultrasonic-sensors-how-they-work-and-how-to-use-them-with-arduino) (where D is the distance, T is the time, and C is the speed of sound ~ 343 meters/second).



1. **Serial Communication Port**

A serial port connection can be used for inter-processor communication within a system or for communication with different parts of a system. The serial port provides the physical connection between the equipment but a communication protocol has to be used to ensure a reliable, error-free data path.

We use Serial Communication Monitor (UART1 Terminal in Keil) to observe the simulation and output of our code.

**SOFTWARE USED**

1. **Keil IDE**

[Keil MDK](https://www.keil.com/mdk5) is the complete software development environment for a range of Arm Cortex-M based microcontroller devices. MDK includes the [µVision IDE](https://www.keil.com/uvision) and [debugger](https://www.keil.com/debug), [Arm C/C++ compiler](https://www.keil.com/mdk5/compiler/6), and essential [middleware](https://www.keil.com/middleware) components. It supports all silicon vendors with [more than 9,500 devices](https://www.keil.com/dd2) and is easy to learn and use.

1. **Proteus**

The Proteus Design Suite is a proprietary software tool suite used primarily for [electronic design automation](https://en.wikipedia.org/wiki/Electronic_design_automation). The software is used mainly by electronic [design engineers](https://en.wikipedia.org/wiki/Design_engineer) and technicians to create [schematics](https://en.wikipedia.org/wiki/Schematic) and electronic prints for manufacturing [printed circuit boards](https://en.wikipedia.org/wiki/Printed_circuit_board).

**ALGORITHM / FLOWCHART**

1. Start. Press Switch1
2. Send Trigger High pulse (delay) for 10us
3. Trigger low
4. Echo High - start timer
5. Wait for the reflected signal
6. Echo pin low and STOP Timer
7. Take value in T0TC in echoTime variable
8. Calculate distance1 (initial distance)
9. Press Switch2
10. Repeat from steps 2-7
11. Calculate distance2 (final distance)
12. distance = distance1 - distance2
13. Calibrate distance to liter using linear equation y=mx +c.

(where calculated values for m=431.3674714 and c= -345.134817 )

1. Convert this float liter to character
2. Send calculated value over Serial Communication port UART0 to display the amount of petrol actually poured into the tank
3. Stop

**SOFTWARE CODING**

**main.c**

#include <lpc214x.h>

#include<stdio.h> //visit http://www.ocfreaks.com/retarget-redirect-printf-scanf-uart-keil/

#include "lib\_funcs.h" //OCFreaks LPC214x Tutorials Library Header

#include "StringUART0Trans.c"

#include "FloatToString.c"

#include "doubleToString.c"

#define TRIG (1<<2) //P0.2

#define ECHO (1<<3) //P0.3

int main(void)

{

initUART0(); //Initialize UART0 for retargeted printf()

UART0\_String\_Tx(" Serial Initialized : ");

initClocks();

//Set PCLK = CCLK = 60Mhz - used by: UART, Timer and ADC

initTimer0(); //Init Timer for delay functions

int echoTime=0;

float distance1;

unsigned char dist1;

unsigned char finalltrs;

IO0DIR |= TRIG; //Set P0.2(TRIG) as output

IO0DIR &= ~(ECHO); //Set P0.3(ECHO) as input (explicitly)

IO0CLR |= TRIG; //Set P0.2 LOW initially

while(1)

{

//Output 10us HIGH on TRIG pin

IO0SET |= TRIG;

delayUS(10);

IO0CLR |= TRIG;

while(!(IO0PIN & ECHO)); //Wait for a HIGH on ECHO pin

startTimer0(); //Start counting

while(IO0PIN & ECHO); //Wait for a LOW on ECHO pin

echoTime = stopTimer0()/10000; //Stop counting and save value(us) in echoTime

distance1 = (echoTime)/58.20; //Find the distance

UART0\_String\_Tx("\n Distance (in cm) : ");

//initial\_level\_inltrs = (0.00231564 \* distance1) + 0.80702712;

//UART0\_String\_Tx(" You have ");

//UART0\_String\_Tx(initial\_level\_inltrs);

delayMS(100);

dist1= float\_to\_string(distance1);

for( int i=0;i<7;i++)

{

uartWrite(r[i]);

}

float finalliters = (431.3674714 \* distance1) - 345.134817;

finalltrs = float\_to\_string(finalliters);

UART0\_String\_Tx("\t Litres (in ml) : ");

for( int i=0;i<7;i++)

{

uartWrite(r[i]);

}

//}

delayMS(1000); //wait 1 second for next update

}

}

// distance = distance2 - distance1;

// final\_level\_inltrs = (0.00231564 \* distance) + 0.80702712;

**lib\_funcs.c**

/\*(C) Umang Gajera- www.ocfreaks.com

Support Library for OCFreaks LPC214x Tutorials.

More Embedded tutorials @ www.ocfreaks.com/cat/embedded/

License: GPL.\*/

#include <lpc214x.h>

#include <stdio.h>

#include "lib\_funcs.h"

//Retarget printf to divert output to UART0

//visit http://www.ocfreaks.com/retarget-redirect-printf-scanf-uart-keil/ for complete printf retargetting tutorial

struct \_\_FILE

{

int dummyVar; //Just for the sake of redefining \_\_FILE, we won't we using it anyways ;)

};

FILE \_\_stdout; //STDOUT

FILE \_\_stdin; //STDIN

int fputc(int c, FILE \* stream)

{

uartWrite(c); //Transmit Character

return c; //return the character written to denote a successful write

}

int fgetc(FILE \* stream)

{

char c = uartRead(); //Receive Character

uartWrite(c); //To echo Received characters back to serial Terminal

return c;

}

void uartWrite(char c)

{

while (!(U0LSR & THRE)); //wait till the THR is empty

if( c == '\n' ) //Send <CR+LF>

{

U0THR = CARRIAGE\_RETURN;

while(!(U0LSR & THRE));

U0THR = LINE\_FEED;

}

else

{

U0THR = c;

}

}

char uartRead(void)

{

while(!(U0LSR & RDR)); // wait till any data arrives

return U0RBR;

}

void initTimer0(void)

{

//Assuming PCLK = 60Mhz

T0CTCR = 0x0;

T0PR = 60-1; //60 clock cycles @60Mhz = 1 us

T0TCR = 0x02; //Reset Timer

}

void delayUS(unsigned long microseconds) //Using Timer0

{

T0TCR = 0x02; //Reset Timer

T0TCR = 0x01; //Enable timer

while(T0TC < microseconds); //wait until timer counter reaches the desired delay

T0TCR = 0x00; //Disable timer

}

void delayMS(unsigned int milliseconds)

{

delayUS(milliseconds \* 1000);

}

void startTimer0(void)

{

T0TCR = 0x02; //Reset Timer

T0TCR = 0x01; //Enable timer

}

unsigned int stopTimer0(void)

{

T0TCR = 0x00; //Disable timer

return T0TC;

}

void initUART0(void)

{

//Assuming PCLK = 60Mhz

PINSEL0 = 0x5; /\* Select TxD for P0.0 and RxD for P0.1 \*/

//U0LCR = 3 | (1<<7) ; /\* 8 bits, no Parity, 1 Stop bit | DLAB set to 1 \*/

U0LCR = 0x80;

//U0DLL = 110;

//U0DLM = 1;

U0DLM = 0x00;

U0DLL = 0x62;

U0FDR = (MULVAL<<4) | DIVADDVAL; /\* MULVAL=15(bits - 7:4) , DIVADDVAL=0(bits - 3:0) \*/

//U0LCR &= 0x0F; // Set DLAB=0 to lock MULVAL and DIVADDVAL

U0LCR = 0x03;

//BaudRate is now ~9600 and we are ready for UART communication!

}

//---------PLL Related Functions :---------------

//Using PLL settings as shown in : http://www.ocfreaks.com/lpc214x-pll-tutorial-for-cpu-and-peripheral-clock/

void setupPLL0(void)

{

//Note : Assuming 12Mhz Xtal is connected to LPC2148.

PLL0CON = 0x01;

PLL0CFG = 0x24;

}

void feedSeq(void)

{

PLL0FEED = 0xAA;

PLL0FEED = 0x55;

}

void connectPLL0(void)

{

while( !( PLL0STAT & PLOCK ));

PLL0CON = 0x03;

}

void initClocks(void)

{

setupPLL0();

feedSeq(); //sequence for locking PLL to desired freq.

connectPLL0();

feedSeq(); //sequence for connecting the PLL as system clock

//SysClock is now ticking @ 60Mhz!

VPBDIV = 0x01; // PCLK is same as CCLK i.e 60Mhz

//PLL0 Now configured!

}

**FloatToString.c**

/\*\*\* Convert float to string \*\*\*/

unsigned char r[6];

int n\_tu(int number, int count)

{

int result = 1;

while(count-- > 0)

result \*= number;

return result;

}

unsigned char float\_to\_string(float f)

{

long long int length, length2, i, number, position, sign;

float number2;

sign = -1; // -1 == positive number

if (f < 0)

{

sign = '-';

f \*= -1;

}

number2 = f;

number = f;

length = 0; // Size of decimal part

length2 = 0; // Size of tenth

/\* Calculate length2 tenth part \*/

while( (number2 - (float)number) != 0.0 && !((number2 - (float)number) < 0.0) )

{

number2 = f \* (n\_tu(10.0, length2 + 1));

number = number2;

length2++;

}

/\* Calculate length decimal part \*/

for (length = (f > 1) ? 0 : 1; f > 1; length++)

f /= 10;

position = length;

length = length + 1 + length2;

number = number2;

if (sign == '-')

{

length++;

position++;

}

for (i = length; i >= 0 ; i--)

{

if (i == (length))

r[i] = '\0';

else if(i == (position))

r[i] = '.';

else if(sign == '-' && i == 0)

r[i] = '-';

else

{

r[i] = (number % 10) + '0';

number /=10;

}

}

}

**StringUART0Trans.c**

#include "lib\_funcs.h"

void UART0\_String\_Tx(unsigned char \*serial\_ptr)

{

unsigned int i;

for(i=0; serial\_ptr[i] != '\0'; i++)

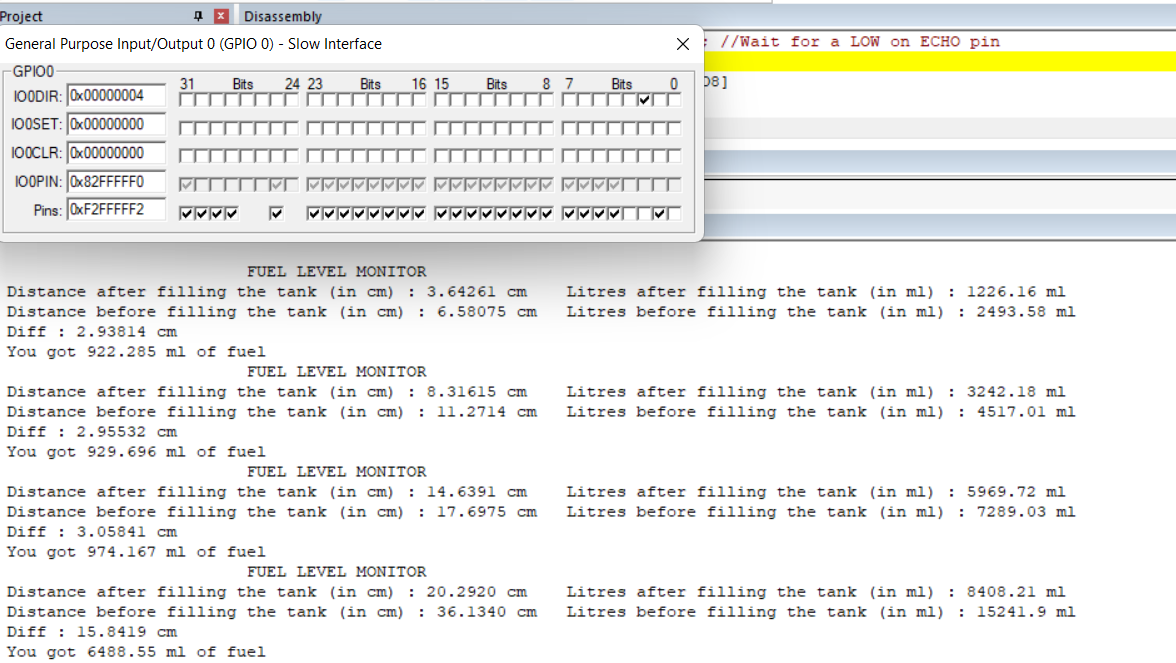
{

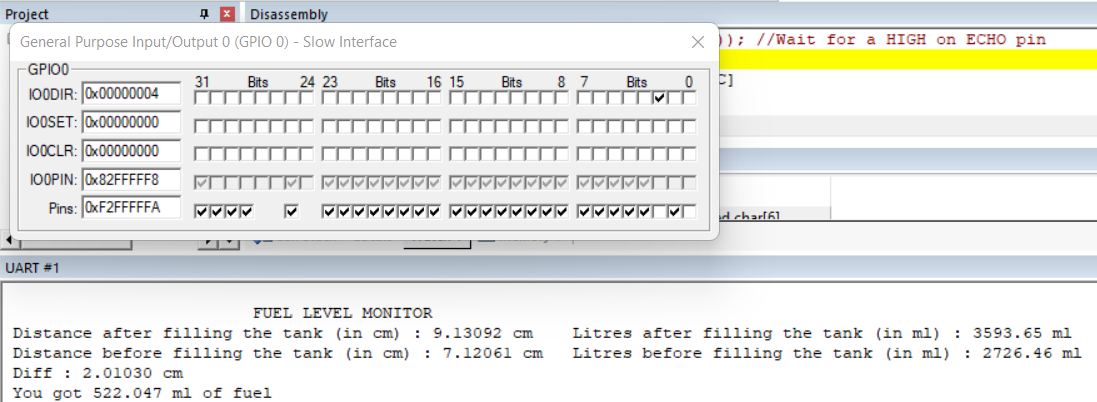
uartWrite(serial\_ptr[i]);

}

}

**OUTPUT**

****



**APPLICATION & ADVANTAGE**

**Application-**

Monitoring the level of the fuel tank serves as a tool for -

* Measure precise volumes of fuel tank refill;
* Detect fuel siphoning from fuel tank;
* Prevent fuel theft from tank (e.g. underfilling or siphoning);

**Advantage-**

* Higher transparency of fuel costs accounting
* Eliminating fuel theft from tank
* Extended lifetime of fuel economy and optimize fuel costs
* Increased fuel economy and optimized fuel cost
* Easier trip reporting and transfer of vehicle between drivers

# 

# CONCLUSION

We have successfully designed a fuel monitoring system and in turn providing a solution for customers to stop being conned by huge petrol stations by not receiving the exact amount of fuel they have already paid for.

Customers can easily identify the discrepancy between the amount shown on the gas station gauges and the real calibrated fuel quantity owing to our system's capacity to show both the initial fuel level in the fuel tank and the fuel level upon adding the fuel.

Our method is straightforward, easy to use, and effective at estimating how much fuel the consumer will actually receive. It also provides users with an overview of how large stations defraud unsuspecting customers by fooling them into paying more than the product is worth. Since this technology is more accurate than any conventional ways of measuring fuel level, users can successfully avoid all such stations.