Design, Development, and Prototyping of an Intelligent Volumetric Measurement System for Water Containers

Minoo Ekrani

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
February 2015
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Asst. Prof. Dr. Mehmet Bodur
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Muhammed Salamah
2. Asst. Prof. Dr. Adnan Acan
3. Asst. Prof. Dr. Mehmet Bodur

ABSTRACT

The aim of this thesis is to build an intelligent water volume measurement device that calculates the water volume for any shape of tank precisely by measuring the water level with an ultrasonic sensor and using a look-up-table to calculate the volume. It generates alarm signals at any desired volume. For any shape of tank, the look-up-table is entered to the processors memory using five buttons, and LCD display. The prototype gets power from mains through a USB socket even if it is not connected to a computer. The system is designed at a system-level using basic building blocks for Arduino from a Arduino Uno processor board, a keypad + LCD display shield with a LED, an ultrasonic HC-SR04 sensor unit, and an external buzzer. The prototype is tested successfully for a horizontally placed cylindrical container to have maximum 1.3%, average 0.7% error on distance and percent volume measurements.

Keywords: Percent volume meter, Ultrasonic volume measurement, system level design.

ÖZ

Bu tez herhangi biçimli bir tanktaki su seviyesini ultrasonik algılayıcı ile hassaslıkla ölçerek bir tablo aracılığıyla su hacmini hesaplayan akıllı bir su hacmi ölçüm sistemi tasarlamayı ve prototipini üretmeyi amaçlamaktadır. Aygıt istenen bir hacimda alarm sinyali üretmektedir. Hacim hesap tablosu tankın biçimine göre işlemcinin belleğine bir LCD ve beş düğme aracılığıyla girilebilmektedir. Prototip, gücünü bilgisayara bağlı olmasa bile USB soket aracılığı ile şebekeden almaktadır. Aygıt Arduino-Uno işlemci kartı, LCD gösterge kartı, bir ultrasonik HC-SR04 algılayıcı birim ile bir harici alarm-sesi üretecinden oluşmaktadır. Prototip yapılan testte yatay yerleştirilmiş bir silindirik kaptaki kalan su hacmini en yüksek 1.3%, ortalama 0.7% hata ile ölçmeyi başarmıştır.

Anahtar Kelimeler: Hacım yüzdesi ölçer, Ultrasonik hacım ölçümü, Sistem düzeyinde tasarım.

To My Family

ACKNOWLEDGMENT

At first, I thank God for giving me the strength and potential to continue, guidance and opportunity to pursue the present study up to a concluding level. Secondly, I would like to express my special appreciation and thanks to my advisor Asst. Prof. Dr. Mehmet Bodur, who has been a tremendous mentor for me. I would like to thank him for encouraging my research and for promoting me to become a researcher. His priceless pieces of advice in my research have always been my guidelines.

I would also like to thank my committee members, Asst. Prof. Dr. Adnan Acan and Assoc. Prof. Dr. Muhammed Salamah.

My special thanks go to my family. Words cannot express how grateful I am to my mother, father and my brothers for all of the sacrifices they have made on my behalf. Your prayer for me was what sustained me thus far.

I would also like to thank all my friends who supported me in writing, and motivated me to strive towards my goal.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Overview

Water containers of unexpected shapes are used for domestic and agricultural purposes as well as in industry. Measuring the water volume in a solid container of unknown geometric shape is possible if the shape of container and the height of water must be known. In this thesis, an electronic device has been designed and prototyped to calculate the volume of the water from the height of the water for any shape of container. The prototype has been tested on a horizontal cylinder tank with 100 cm diameter.

1.2 Microcontrollers

A microcontroller is a single chip computer complete with RAM, ROM, I/0 units, and several land of peripheral units. There are many microcontroller families targeting 8-bit, 16-bit and 32-bit embedded system applications. Some examples of well known microprocessor families are Microchip PIC16, PIC18, and PIC30 families, Atmel AVR 8-bit and 32-bit families (8051 compatible ATMega, and 32 bit XMega families), and Motorola 68F families. Although most of these families have almost equivalent members in price and performance, Intel 8051core has been widely known in engineering education because of wide support available by the manufacturing companies such as ATMEL, Zilog, Honeywell, Fairchild, etc.

1.3 Embedded System Development Boards

The development of surface mount technology increased number of available pins on a device, and reduced the cost of the circuit boards considerably. Many microcontroller manufacturers started to developed microcontroller boards with many features and functions so that their products can be interfaced directly to a PC, with benefits of fast development cycles of embedded systems. Similar to almost all microcontroller manufacturers, Atmel produced and widely distributed ATMega-8 development boards, including to the Electrical and Computer Engineering departments for promoting common use of their products. Availability of these evaluation boards affected the perspective of embedded design projects, leading them to a new system level approach instead of building them component by component from the zero level.

1.4 Arduino-UNO as a Microcontroller Board

The core processor 8051 has been introduced by Intel in 1980, and for long years it is used in BS and MS level design projects along with the industrial applications. ATMEL and Intel supported this core processor by distributing educational purpose assemblers, and compilers. As a result of this wide support, several tools were developed for ATMega 8051 core family including the Code Blocks IDE compatible C and C++ compilers. After many years of educational support, some universities modified ATMega evaluation boards for their common project requirements, and developed a family of system level microcontroller boards, with many plug-in add-on boards.

Arduino UNO is one of these development boards, based on ATMega-8 family processors. The board includes:

i) A pre-programmed microcontroller for programming through serial port,

ii) Some necessary components for its stable and reliable operation such as a crystal clock and a voltage regulator,

iii) A USB bus to serial port converter for a convenient communication by a PC,

iv) Some sockets to connect it to other peripheral boards, which are called shields.

The wide usage of Arduino boards in educational projects reduced their price to less than $10. Many interfacing projects for Arduino boards enriched the available shields and sensors over a hundred boards.

1.5 Sensors for Water Level Measurement

Water level detection is a necessity for many automatic process control systems. Contact methods are mostly simple and non-expensive, such as:

i) A float switch made of a light material that floats on the water surface at the end of a cable, which is connected to a mercury switch;

ii) A float and lever mechanism that moves a potentiometer for continuous level monitoring.

There are non-contact methods that use:

i) Pressure difference by the height of water, –ii) capacitive or inductive effect developed by the water,

ii) Ultrasonic measurements using a ultrasonic transmitter and receiver to measure the empty distance of the vessel.

Among these methods, the float switch is used conventionally for almost all automatic pump control systems in domestic clean and dirty water systems, because although the sensor needs contact the float and cable can be constructed using safe materials for water-proof construction for a couple of dollars. However, one of the driving motivations of this thesis was the availability of all system level blocks to construct an intelligent ultrasonic based system at competing prices to the water-proof float switches.

Mainly, HC-SR04 Ultrasonic-sensor is made of a 40 kHz transmitter and receiver piezo-electric ceramic discs with a metal uni-cone horn to focus the ultrasonic beam produced by Ceramic Transducer Design Co., Ltd. in Taiwan since 30 years. The transmitter requires high driving voltage, which is mostly generated by a MAX232 line driver chip. The receiver is interfaced to the input of an active multi-stage band-pass filter made of LM324. The transmitter and receiver circuits were interfaced to the trig and echo terminal pins by a 14 pin STC11 (8051 core) microcontroller.

1.6 User Interface Devices

An LCD display module contains all necessary control circuitry to drive a dot matrix LCD glass so that it can be interfaced to a microcontroller easily by either 8 or 4 data lines and 3 control lines. In this thesis, the measured distance from the sensor to water surface. The calculated percent water volume is displayed on LCD module.

The system-level-design approach of Arduino-UNO project provides a well-known commercial LCD display with a keypad shield to be plugged directly on the Arduino-UNO board. This unit has total 6 button switches and a 2-line by 16 character LCD module on the shield.

Other than the LCD-shield, the design aims to accomplish audio-visual warnings for the low water levels through a LED, and a buzzer. The LED output may also be converted to a relay output using a transistor to drive the relay coil. The relay contacts may provide sufficient power for pumps and other similar equipment, which are necessary for the automatic control of the water reservoir.

1.7 Design and Prototyping Phases of the Developed System

This thesis aims the design, implementation, and prototyping of an intelligent water level and volume measurement system by a systems level approach using an ultrasonic sensor to measure the water level in a tank. The thesis targeted the following phases:

The first phase is design of an intelligent water level measurement that use ultrasonic sensor to measure the water volume in a water container by system level approach instead of component level approaches. It shall be suitable to install well-developed interface units such as relay, LCD display and buttons RS232 and network interface as add-on units to a 8051 family processor board such as Arduino-UNO with a USB port connection to a computer. The code of the system may be developed on an Arduino compatible IDE, which allows easy debugging of the code without prototyping.

The second phase is the prototyping of the design for a cylindrical tank with a 100 cm diameter and 110 cm height. The prototype shall be re-programmable through the USB connection, but shall work standalone without any connection to a computer port.

The third phase is to test the prototype on actual working conditions with a water container, volume from the output of the height, and show it on LCD.

Chapter 2

HARDWARE OF THE VOLUMETRIC MEASUREMENT SYSTEM

This Chapter explains the main parts of the water level sensor system, and introduces an overview of the literature for similar systems. It explains the characteristics of components that they used in Water Level Measurement and how they work.

2.1 Hardware Components Used in Designing the System

The system is mainly composed of available building blocks in the market such as:

i) Arduino Uno processor,

ii) an ultrasonic distance sensor,

iii) an LCD display shield with a keypad for Arduino.

iv) A buzzer and an LED are added to the system to give out visual and audio warning signals for low and high level of water.

2.2 Introduction to Arduino Hardware

Arduino was developed in 2005 by Prof. Massimo Banzi, in Ivrea, Italy, for student projects as a replacement of "BASIC Stamp" that costs $100, which is too expensive for students. The name "Arduino" comes from “Arduino, King of Italy from 1002 to 1014”. Colombian student Hernando Barragan contributed Arduino hardware by his MS Thesis on a wiring platform. Many contributors worked on it to make it less expensive, and easy to use by the open source community.

The Arduino-UNO board is based on an Atmel 8-bit AVR microcontroller, mostly running by a 16 MHz crystal oscillator. The board contains a 5V regulator, and many peripheral parts and a microcontroller with a self-programming pre-loaded code. The i/o pins are connected to a set of standard connectors, that allows connecting many extra peripheral boards, which is called *shields*, on top of the processor boards. The shields were designed under the Arduino open-source project, for common use in MS and BS projects. Many of these shields have I2C-bus, which allows problem free connection of multiple shields to a single Arduino-UNO. The boot-loader in the pre-programmed Arduino microcontroller provides uploading of program codes from the Arduino IDE to the flash-ROM program memory of the controller by an RS232 serial connection. In the recent design of the Arduino-UNO boards, a USB interface is available through a RS232-USB converter on the board.

From total 20 i/o pins, 6 of them may be used for analog input as well as digital i/o. Another 6 pins can generate pulse-width-modulated (PWM) output as well as digital i/o. PWM outputs may be converted to an analogue output by only filtering the pulses with an RC filter. A number of plug-in shields are commercially available for a system level design of any microcontroller project. The assignment of microcontroller pins to the Arduino-UNO board is shown in Figure 2.1.

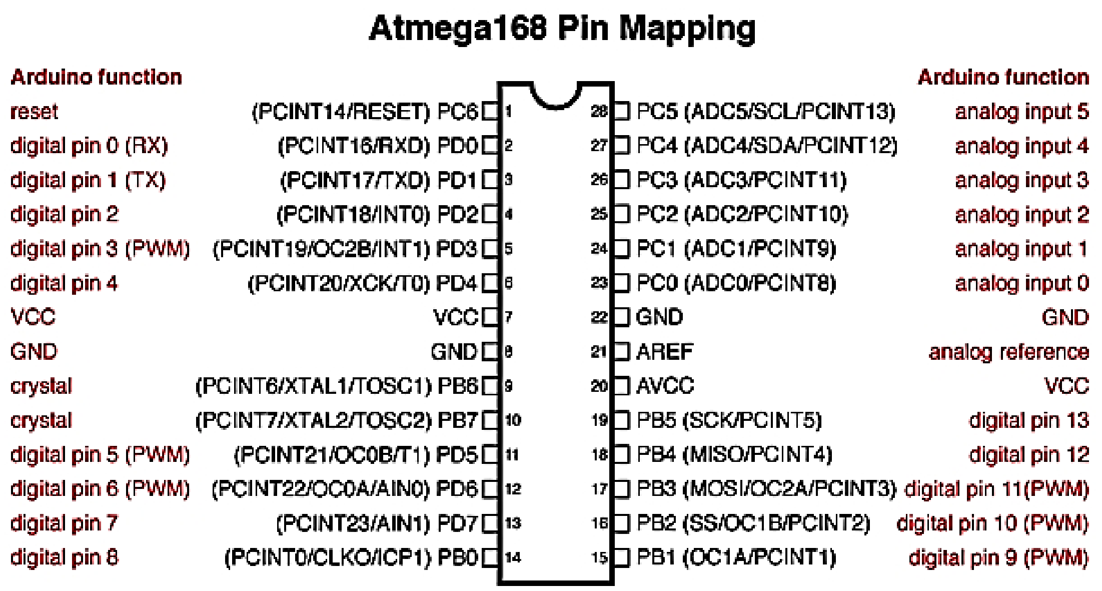


Figure 2.1: Assignment of microcontroller pins to terminals of the Arduino Board

2.3 Arduino Software Platform

Arduino Integrated Environment was started to develop a programming medium for common wiring dedicated to student projects. Further developments of the language and medium resulted in today available Integrated Development Environment, which is implemented in Java programming language.

Arduino IDE was developed mainly for the students with limited programming knowledge. Arduino IDE code is called a "sketch". The code editor has advanced features such as syntax highlighting, automatic indentation, brace matching. A click of mouse is sufficient to compile and upload programs to an Arduino Board.

Arduino IDE codes are written in C or C++. Many library routines come with "Wiring" library, which are written before the development of Arduino IDE. A program that runs in an endless loop is constructed by two functions: setup(), and loop().

setup() is called at the beginning of the program to initialize the settings such as i/o ports and peripherals.

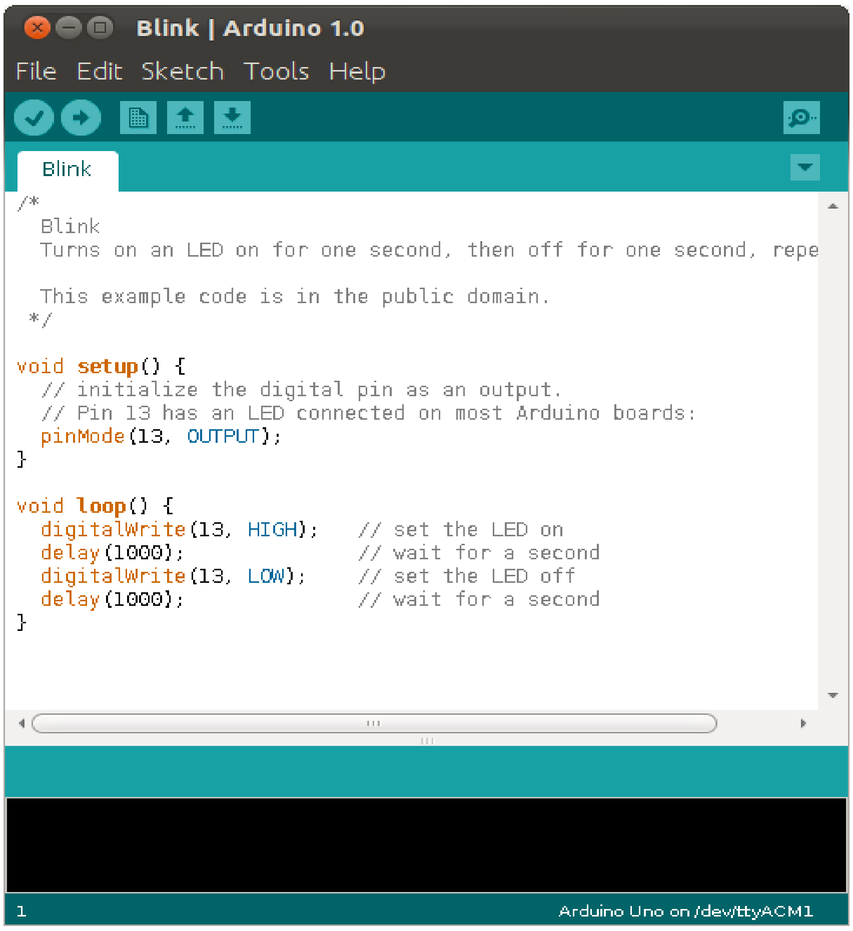
loop() is a function that generates a loop with the declared contents. The content of loop function is repeatedly executed in an endless loop.

A typical "hello" program for a microcontroller system is a flashing LED. Arduino boards are installed with a LED and resistor from pin-13 to ground (GND). It is blinked by the following sketch, which is mainly used to test the Arduino Board quickly.

```
#define LED_PIN 13
void setup () {
pinMode (LED_PIN, OUTPUT); // Enable pin 13 for digital output
}
void loop () {
digitalWrite (LED_PIN, HIGH); // Turn on the LED
delay (1000); // Wait one second (1000 milliseconds)
digitalWrite (LED_PIN, LOW); // Turn off the LED
delay (1000); // Wait one second
}
```

The sketch code for Arduino is compiled only in Arduino IDE environment, which converts the sketch to C or C++ code, and calls the Atmel C - C++ compiler. The icon "Upload code to i/o board" starts compilation, and transfers the executable to the Arduino board.

Arduino platform uses Atmel microcontrollers, Atmel's development environment, AVR Studio or the newer Atmel Studio. These platforms are compatible to develop software for the Arduino.



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

```
1                                        Arduino Uno on /dev/ttyACM1
```

Figure 2.2: Arduino 1.0, on Ubuntu Linux 11.10

2.4 LCD Keypad Shield

LCD Keypad shield is a convenient add-on for Arduino board to provide a simple user-friendly input-output. It combines 6 push buttons and a 2x16 LCD display. The status of push-buttons are converted to analogue voltage by a resistor network, and read by A0 analogue pin. Data interfacing is carried out by pins 4, 5, 6, 7, 8, 9 and 10. The LCD shield has a trim-pot for contrast adjustment and a backlit on/off circuit. With a 5-button keypad, it provides an interface for a programmable menu to implement a user-friendly input-output method. The keys are usually called "select", "up", "right", "down" and "left".

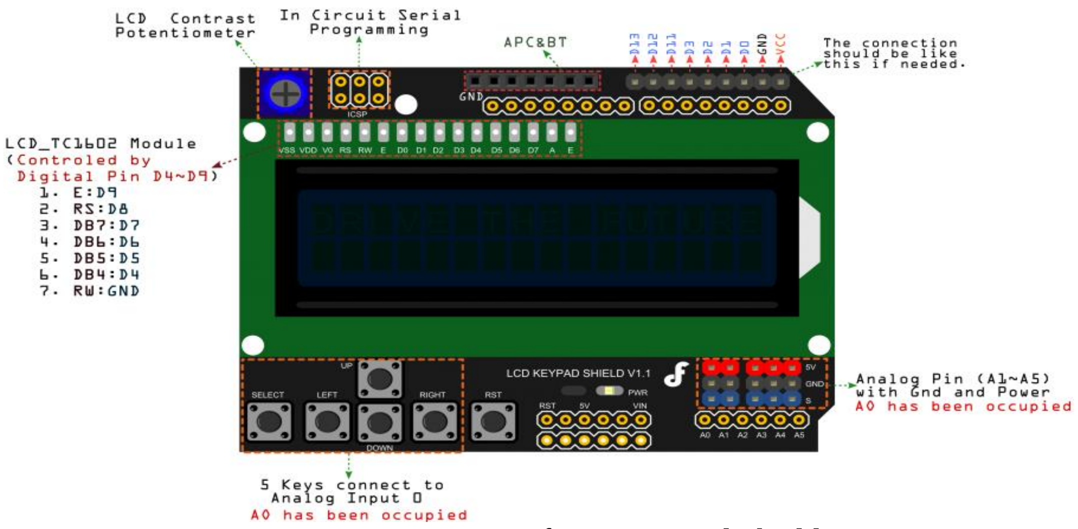

LCD Contrast Potentiometer
In Circuit Serial Programming
APC&BT
D13 D12 D11 D3 D2 D1 D0 GND VCC
The connection should be like this if needed.
LCD_TC1602 Module (Controled by Digital Pin D4~D9)
1. E:D9
2. RS:D8
3. DB7:D7
4. DB6:D6
5. DB5:D5
6. DB4:D4
7. RW:GND
ICSP
GND
VSS VDD V0 RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A E
LCD KEYPAD SHIELD V1.1
PWR
RST 5V VIN
5V GND S
A0 A1 A2 A3 A4 A5
UP SELECT LEFT RIGHT RST DOWN
Analog Pin (A1~A5) with Gnd and Power
A0 has been occupied
5 Keys connect to Analog Input 0
A0 has been occupied

Figure 2.3: Diagram of LCD Keypad Shield

Table 2.1: Terminals of LCD Keypad Shield

Pin	Function	Explanation
D4	D4 ~ D7 are data lines DB4 ~ DB7	Bidirectional tristate data bus. Transfers data between the MPU and the LCD.
D5		
D6		
D7		
D8	RS	Signal Display or Choose Data
D9	Enable	Starts data write / read
D10	Controls LCD Backlight	
A0 (Analog)	Reads Pressed Button	Select, down, up, left, and right

The following paragraphs explain available LCD Functions:

LiquidCrystal(rs, enable, d4, d5, d6, d7) is a procedure that assign the i/o pins to an initialized LiquidCrystal variable. The display can be interfaced by either 4, or 8 data lines. If it is connected by 4 lines, you shall omit the pin numbers from d0 to d3 leaving them unconnected. The RW pin is omitted by connecting it to the ground. If you omit RW, you shall remove it from the parameters of the function.

lcd.begin(cols, rows) is a procedure that initializes an LCD screen, and specifies the number of characters in a line (cols), and number of lines (rows). lcd.begin() shall be called before all other LCD library commands.

lcd.setCursor(col,row) sets the location where subsequent text is displayed on LCD.

lcd.print(data) displays text on the LCD.

lcd.write(data) writes a single character to the LCD.

2.5 Ultrasonic Distance Sensor

The HC-SR04 ultrasonic sensor uses reflected ultrasonic sound waves at 40 kHz to determine the distance to an object from the fly time of the sound. It is a high accuracy non-contact distance sensor with stable readings with transmitter and receiver modules in a convenient package. Measurement accuracy is not affected by light or darkness of target. The only disadvantage is the difficulty of detection of soft materials like cloth that does not reflect ultrasonic waves.

Ultrasonic sensor is used for distance measurement up to 4 meters, at 100ms periods. It is suitable for applications with distance measurement where target moves slowly or stays stationary.

Interfacing to a microcontroller is very simple. It needs a 10 microsecond trigger pulse to produce 8 pulses of ultrasonic waves at 40 kHz. Another pin sends a pulse during the fly period of “echo”. The distance is proportional to the duration of this pulse. The pulse width, T_p, is proportional to the distance, *d*. by proportionality factor which is related to the speed of the ultrasonic waves in the air.

Table 2.2: Key Features of Ultrasonic Sensor

PowerSupply	+5V DC
QuiescentCurrent	<2mA
WorkingCurrent	15mA
EffectualAngle	<15°
RangingDistance	2cm – 400cm/1"-13ft
Resolution	0.3 cm
MeasuringAngle	30 degree
TriggerInputPulsewidth	10µS
Dimension	45mm x20mm x15mm

There is a LED on the sensor board to indicate the echo pulse. The sensor board has an 4-pin header as shown in Table 2.3. Connection does not need soldering if you use this header, directly to a controller board or through an extension cable.

Table 2.3: Pin Layout of Sensor Socket

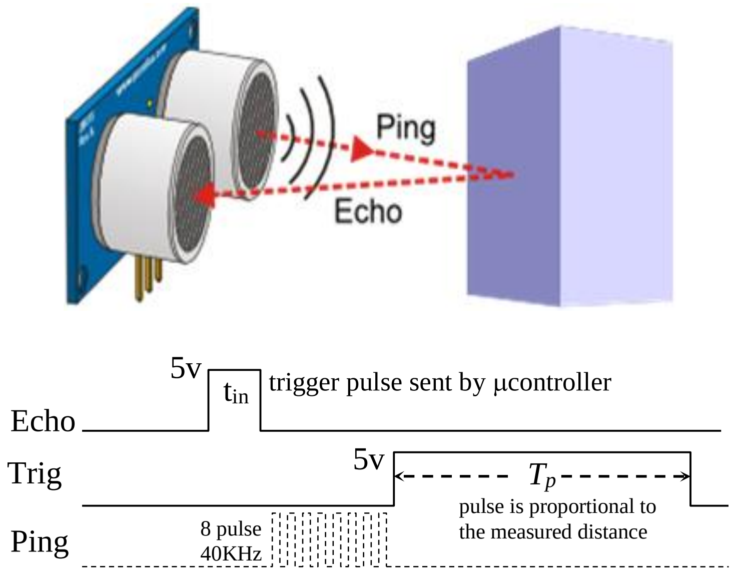
VCC	**Trig**	**Echo**	**GND**
+5VDC	Trigger input of Sensor	Echo output of Sensor	GND

To start measurement, Trig pin of SR04 must receive a 5V pulse for at least 10µs to start the sensor to generate an ultrasonic burst of 8 cycles at 40 kHz. Sensor measures the fly time of the reflected ultrasonic burst until it is detected by the receiver. Then, it sends a positive pulse from Echo pin for a period proportional to the measured fly time. The distance is calculated using the speed of sound in air, which is known almost 340m/s. Since the ping travels twice of the distance, we obtain:

$$d = T_p / 680,$$

where the units of the distance and time shall be meters, and seconds.

The sensor can be connected to any microcontroller with digital i/o such as SK40C, PIC, and Arduino series.


Ping
Echo
5v
t_in
trigger pulse sent by µcontroller
Echo
Trig
5v
Tp
pulse is proportional to the measured distance
Ping
8 pulse 40KHz
Figure 2.4: Graph of Ultrasonic Distance Sensor

2.6 LED Indicator, and Miniature Speaker as a Buzzer

LED (Light Emitting Diode) is a p-n-junction diode, that gives light when driven by a forward current. Electrons that recombine with electron holes at the junction release the gap energy by generating photons, resulting to electroluminescence. The color of the light is related to the gap energy, which may be varied by the substrate and doping elements of the p-n-junction. The size of substrate material of a LED is small, mostly shorter than 1 mm. They are packed in optically transparent cases, to direct the light beam in a desired manner.

Light Emitting Diodes are faster in switching, longer in life, and physically robust compared to a light-bulb. LEDs are currently used in applications such as moving signs, automotive headlamps, general lighting, camera flashes, and lighting in flight.

The LEDs are mainly used in optical distance sensors, and high technology video displays along with lighting of buildings, and roads. In addition, the high switching rates of the LEDs are valuable in advanced correspondences technology.

A buzzer or beeper is an electro-mechanical, or piezoelectric audio signalling device. It is typically used for producing an alarm sound, warning for the timer periods and validation of user input such as a keystroke or a mouse click.

For this design, a high quality internal speaker M3020N004R-0715K is used as a buzzer of the prototype. The size of the speaker is 30 x 20MM, and its main electrical specifications are 4Ω, 1.5W. This speaker has about 1.5W nominal input power, and it is tested for temperatures in the range 0-70 degrees centigrade, at relative humidity 95% for 24 hours.

Figure 2.5: Minature 4 Ohm Speaker used as a Buzzer

Table 2.4: Specifications and Characteristics of M3020N004R-0715K Speaker

Type No:	Internal Speakers for Mobile Phone M3020N004R-0715K
Impedance:	4 ± 0.6Ω (15%) at 1KHz
Input Power:	Nominal Input 1.5W
	Maximum Input 2W
Resonance center Frequency:	390 ± 20% Hz at 1.0V
Magnet Size:	11 x1.5mm (diameter x height)
Operation:	Nominal 1.5W (Music Peak Source)
Voice Coil Diameter:	11.28 mm
DC Coil Resistance:	4 ± 0.28Ω(7%)
Sound Pressure Level: (SPL)	97 ± 3dB at 1W /1M It is acceptable compared with the standard sample± 3dB.
Reliability Test: (Standard test condition under IEC)	Load Test: Must be normal after load test: pink noise with IEC filter 3 W 96 hours (F0-20KHZ)
	Heat Test: Must be normal after 70 ± 2 °C for 24 hours and then room temperature 1 hours
	Cold Test: Must be normal after –25 ± 2 °C for 24 hours and then room temperature 1 hours
	Humidity Test: Must be normal after40 ± 2°C and relative humidity 90-95% for 24 hours then room temperature 1 hours

The speaker needs to be driven by ac voltage to generate a buzzing sound. The ac voltage is generated by pin-3 (PWM output) of the Arduino board at 1kH.

2.7 Explanation of the LCD Shield with a Keypad Circuit

The circuit schematics of the LCD unit with keypad are given in Figure 2.6.

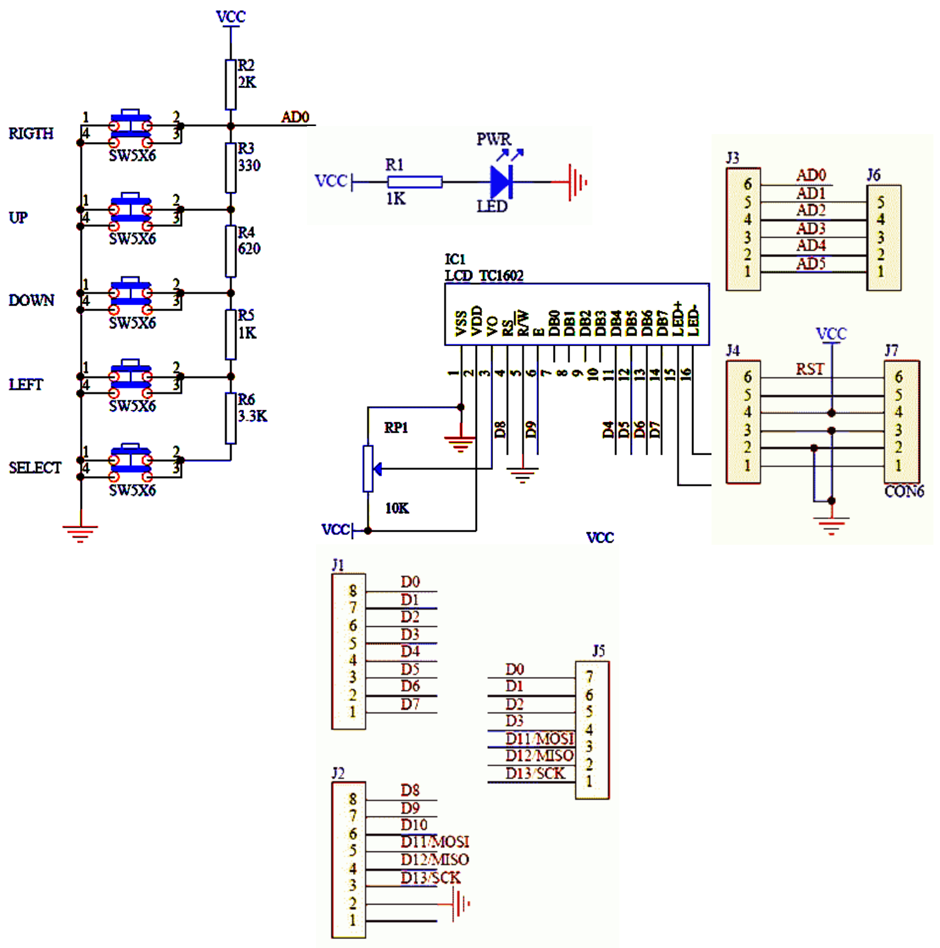


Figure 2.6: Circuit Schematics of LCD Shield with a Keypad

1) Resistors

There are total 6 resistors on this board. The power LED current of the board is restricted by the resistor R1=1k. The 1/4W 5% resistors R2=2k, R3=330 Ohm, R4=620Ohm, R5=1k, and R6=3.3k convert the status of the pushbutton switches to an analogue voltage on AD0. Arduino UNO board reads this voltage using its analogue to digital converter.

2) Potentiometer

There is one 10k trim pot, RP1, for the contrast adjustment of the LCD display. It is near the RESET button.

3) Pushbuttons

There are total 6 pushbutton switches. They are 6mm tactile switch pushbuttons labelled by UP, DOWN, LEFT, RIGHT, SELECT, and RESET on the PCB.

5) Male Header Pins

There are two strips of 36 male header pins on the circuit. These are necessary to attach the shield to the Arduino.

2.8 Overall Design of the System

The system is made of two modules and two components interfaced to an Arduino-UNO microcontroller. The circuit diagram of overall system is given in Figure 2.7.

Pin Definition:

i) PORT B (PB0 – PB7) is an 8 bit bidirectional I/O port with internal pull-ups. Processor pins 14 – 17 bring PB0 to PB5 out.

a. PB0 – PB5 are also interrupts 0-5 respectively.
b. PB1 can also be used as a PWM output.
c. PB2 can also be SPI Bus Master Slave Select (*SS) or PWM output.
d. PB3 can also be or SPI Bus Master Out/Slave In (MOSI) or PWM output.
e. PB4 can also be SPI Bus Master In/Slave Out (MISO).
f. PB5 can also be SPI Bus Master Clock Input (SCK).
g. PB6 and PB7 are brought out on Processor pins 9 and 10 for the crystal clock oscillator.

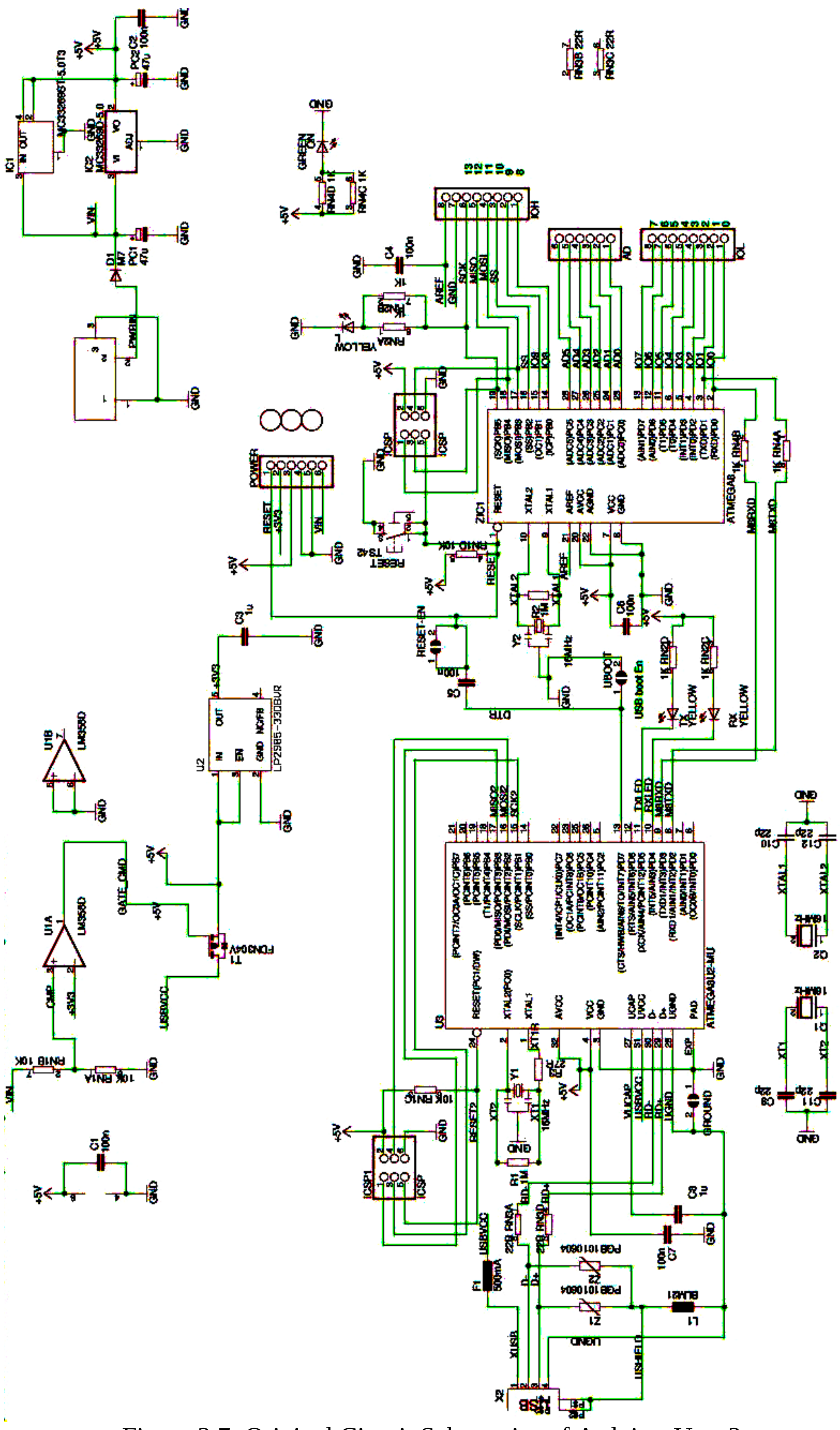


Figure 2.7: Original Circuit Schematics of Arduino-Uno-3

ii) PORT C (PC0 – PC5) is a 7 bit bidirectional I/O port with internal pull-up resistors. Processor pins 23 – 28 bring PC0 to PC5 out.

a. PC0 – PC5 are also interrupts 8-13 respectively.
b. PC0 – PC5 can also be used as A/D inputs.
c. PC4 and PC5 can also be used as SDA and SCL for I2C.
d. PC6 is brought out on processor pin 1 as reset.

iii) PORT D (D0 – D7) is an 8 bit bidirectional I/O port with internal pull-ups. Processor pins 2 – 6 and 11 – 13 bring all pins out.

a. PD0 can also be USART Input (RXD).
b. PD1 can also be USART Output (TXD).
c. PD3 can also be used as a PWM output.
d. PD5 can also be used as a PWM output.
e. PD6 can also be used as a PWM output.

The Arduino Uno pinout is printed in the silkscreen on the top of the part. At first we will use mainly the pins in the female headers at the edge of the board, plus USB and maybe power.

- Tx and Rx are serial UART pins used for RS-232 and USB communication.
- I2C is a serial communication method using a bidirectional data line (SDA) and a clock line (SCL).
- SPI is a serial communication method using three lines: MOSI (Master Out Slave In), MISO (Master In Slave Out), and a clock (SCK) line.
- A/D is Analogue to Digital conversion. AD pins convert an analogue voltage in to an integer.

- PWM (Pulse Width Modulator) generates a square wave with a specific duty cycle (high period).
- ICSP is the In Circuit Serial Programming – a method to program the processor without removing it from the circuit.
- Vcc is the voltage supplied to the processor (+5VDC regulated from the higher supply input voltage).
- 3.3VDC is regulated from higher-supply-input-voltage to power up some of the peripherals (50ma maximum).
- IOREF provides a voltage reference so shields can select the proper power source.
- AREF is a reference voltage of A/D converters.
- GND is the ground reference.
- RESET resets the processor (and some peripherals).

Chapter 3

IMPLEMENTATION AND PROTOTYPING

3.1 Introduction

This chapter introduces the design of the Water Level Measurement System including the connections of hardware devices, and Arduino software.

Once the system is assembled from the Arduino board, ultra-sonic sensor, and the LCD module, the microcontroller shall be coded with a program to:

a) initialize the ports and shields,

b) start to repeat the following tasks forever:

i. Read the buttons, and if any button is pushed do the necessary action for that button.
ii. Read the distance d, from the sensor to the water surface;
iii. calculate h, the height of water in the container;
iv. display the level and percent volume on LCD module.

The buttons are assigned to the following actions: Select button changes the mode from monitor to setup mode, where Left button increments the index of 20-item Look-up-table, and up-down buttons increments and decrements the contents of the look-up-table. In the setup mode, LCD unit displays the index and the contents of the lookup table.

The program is written in the special Arduino IDE software in C or C++ programming language and it is transferred to the Arduino program memory using Arduino USB port cable. In practice, we applied incremental design procedure, where the system peripheral units are connected to the system one by one. After connecting each unit, we tested the functionality of all units to be sure that the design is satisfactory in all respects.

3.2 Step-By-Step Hardware Assembly

Arduino software has to be installed and set for the correct port to communicate to the Arduino-Uno board. Port configuration and USB connection of the Arduino UNO board is tested by transferring a short program to Arduino that blinks a LED.

Next step is to plug the LCD-keypad shield on the Arduino-Uno, and test the LCD module by writing HELLO on the LCD screen.

The third step is connection of the ultrasonic sensor to the LCD-keypad shield.

An almost 1 meter long 5 line flexible cable is used to connect the ground (GND) of LCD to the ground (GND) pin of the sensor; 5 V of LCD to the Vcc of the sensor. A1 of the LCD to Trig pin of the sensor; and A2 of the LCD to Echo pin of the sensor. By these connections Arduino can start a distance measurement by sending a pulse to its A1 (Trig) pin, and count the duration of the pulse at A2 (Echo) pin to find out the distance in centimeters.

Once the distance *d* from the sensor to the water surface is obtained the water level *h* is calculated by subtracting *d* from the empty tank distance do.

$$h = d_o - d;$$

The water volume is calculated using the distance d to the water surface by piecewise linear function *v* = *f* (*d*). The corner point of the function is stored in a look-up table of 21 elements. The function *v* = *f*(*d*) returns the percent water volume for a given distance *d*.

The percent volume *v* is displayed on the LCD module.

The low volume warning is started if the percent water volume drops below 10%. The alarm status is indicated by a buzzer sound, and by lighting the LED continuously.

The look-up table of the piecewise-linear volume function *v* = *f*(*d*) can be modified even while Arduino is working standalone using the keypad of the LCD module. The developed prototype may measure the volume of the remaining water in any size and shape of containers by modifying the look-up table. The preloaded look-up table values are calculated for a horizontal cylindrical vessel with a diameter dv = 100 cm. The details of the calculations are supplied in Appendix A.

The select button starts the setup mode. Left button displays the next corner point (di, vi) of the function. Up and Down buttons for increase or decrease the value of percent volume.

Figure 3.1 shows the complete prototype of the system. The complete Arduino code of the prototype is listed in Appendix B.

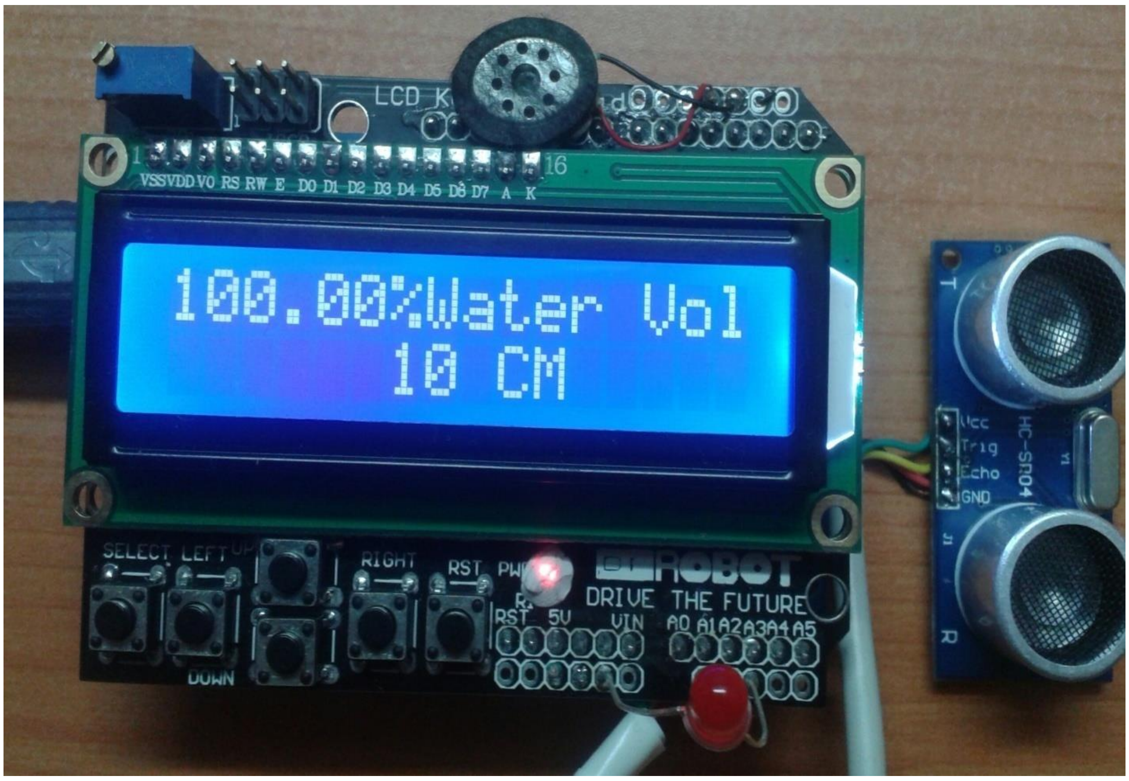


Figure 3.1: Snapshot of the Volumetric Measurement System

3.3 Electrical Connections of Main Building Blocks

The prototype consists of three main building blocks which are commercially available in markets.

a. Connection of Sensor to LCD

The pins of the sensor were connected to GND, 5V, A1, and A2 of the LCD to receive the signal from the controller. GND is connected to GND, Vcc to 5V, Trig to A1, and Echo to A2.

b. Connection of Led to LCD

The pins of the Led were connected to GND, and A3 of the LCD to receive the signal from the controller. (-) is connected to GND, and (+) to A3.

c. Connection of Buzzer to LCD

The pins of the Buzzer were connected to pins number 2 and 3 of digital pins of the LCD to receive the signal from the controller. (-) is connected to pin number 2, and (+) to pin number 3.

Chapter 4

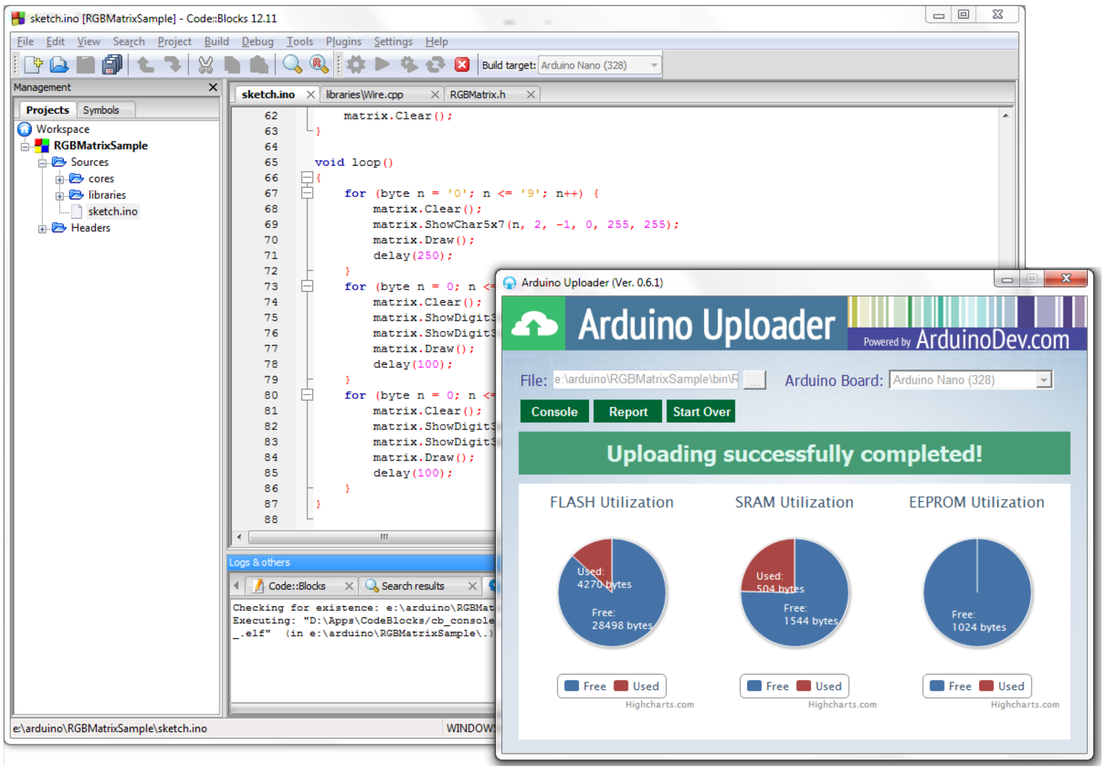
TESTING AND INTERFACING OF THE DEVICES

4.1 Testing of Arduino Program

Instead of developing Arduino software directly on an Arduino board, programs may be developed and simulated on the Code Blocks Arduino Integrated Development Environment (CBA-ID E), which allows the code developer to trace the codes on an Arduino simulator. Simulation of Arduino provides a powerful tool for debugging, and testing the developed codes before building the prototype. In this Thesis, all codes were developed and tested in CBA-IDE environment, to obtain highest coding quality and correctness of the code functionality.

CBA-IDE provides many useful features for code development such as code route, code completion, code folding, accumulating and transferring for Arduino, which simplifies code development projects. The last version (2014) contains very useful tools such as Arduino Builder and compiler, standard Arduino libraries, core files, a serial terminal, AVR tool chain, and, an Arduino simulator at API-level .

The CodeBlocks Arduino IDE allows chip simulation and in-circuit emulation for the Arduino family of microcontrollers. It has an easy to use user interface and its help gives complete information overview. The IDE provides important information to the user by several windows. Some of these windows are the *Workspace*, *Source Code*, *Output*, and *Watch* windows as seen in the Figure 4.1.



```
    matrix.Clear();
}

void loop()
{
    for (byte n = '0'; n <= '9'; n++) {
        matrix.Clear();
        matrix.ShowChar5x7(n, 2, -1, 0, 255, 255);
        matrix.Draw();
        delay(250);
    }
    for (byte n = 0; n <
        matrix.Clear();
        matrix.ShowDigit3
        matrix.ShowDigit3
        matrix.Draw();
        delay(100);
    }
    for (byte n = 0; n <
        matrix.Clear();
        matrix.ShowDigit3
        matrix.ShowDigit3
        matrix.Draw();
        delay(100);
    }
}
```

```
Checking for existence: e:\arduino\RGBMat
Executing: "D:\Apps\CodeBlocks/cb_console
_.elf"  (in e:\arduino\RGBMatrixSample\.)
```

File: e:\arduino\RGBMatrixSample\bin\R

Arduino Board: Arduino Nano (328)

Uploading successfully completed!

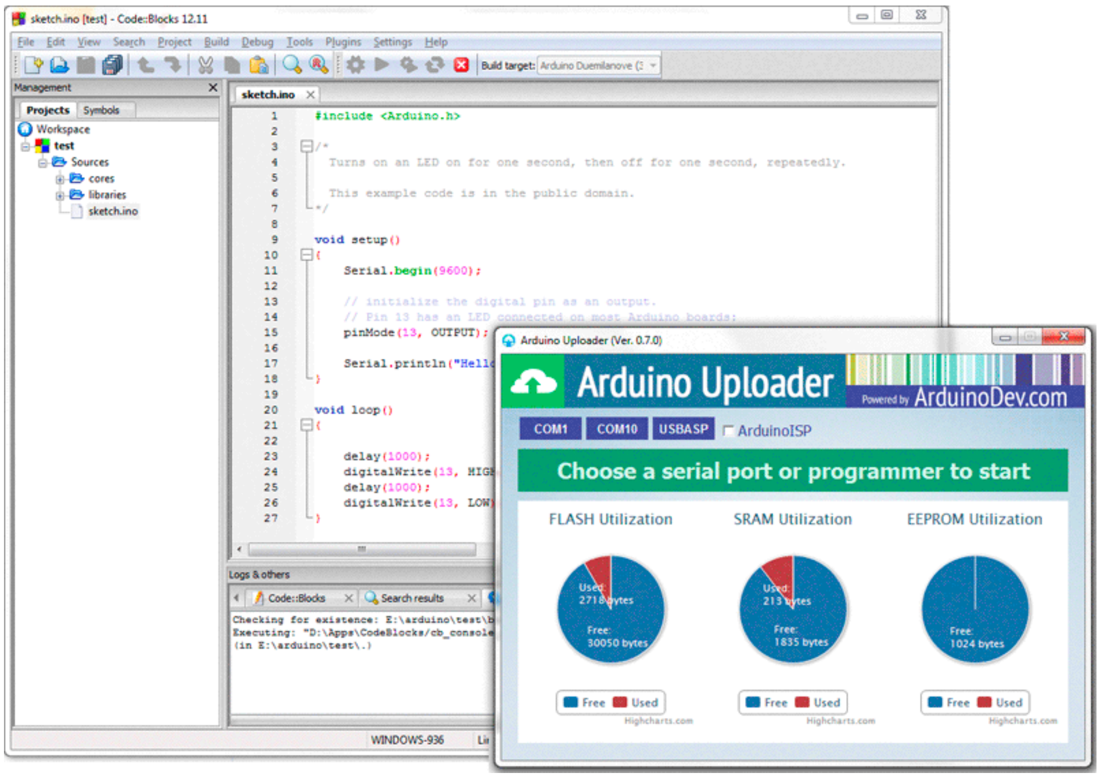
FLASH Utilization: Used: 4270 bytes; Free: 28498 bytes

SRAM Utilization: Used: 504 bytes; Free: 1544 bytes

EEPROM Utilization: Free: 1024 bytes

Figure 4.1: A Sample of IDE Windows when a Program is Simulated

In the box of a following figure we can see that it is asking the right serial port.


sketch.ino [test] - Code::Blocks 12.11
File Edit View Search Project Build Debug Tools Plugins Settings Help
Build target: Arduino Duemilanove (3
Management
Projects Symbols
Workspace
test
Sources
cores
libraries
sketch.ino
Logs & others
Code::Blocks
Search results
WINDOWS-936
Arduino Uploader (Ver. 0.7.0)
Arduino Uploader
Powered by ArduinoDev.com
COM1
COM10
USBASP
ArduinoISP
Free
Used
Highcharts.com

```
#include <Arduino.h>

/*
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
*/

void setup()
{
    Serial.begin(9600);

    // initialize the digital pin as an output.
    // Pin 13 has an LED connected on most Arduino boards:
    pinMode(13, OUTPUT);

    Serial.println("Hell
}

void loop()
{


    delay(1000);
    digitalWrite(13, HIGH
    delay(1000);
    digitalWrite(13, LOW)
}
```

```
Checking for existence: E:\arduino\test\b
Executing: "D:\Apps\CodeBlocks/cb_console
(in E:\arduino\test\.)
```

Choose a serial port or programmer to start

FLASH Utilization: Used: 2718 bytes; Free: 30050 bytes

SRAM Utilization: Used: 213 bytes; Free: 1835 bytes

EEPROM Utilization: Free: 1024 bytes

Figure 4.2: A Snapshot of Choosing a Serial Port

4.2 Results of Water Level Tests

Varying distances with the 9 points differences was given into the sensor and the volume is calculated for the outputs. The results are tabulated in Table 4.1.

In the test, the sensor was placed above a set of distances which corresponds to different water-levels. The height of water is changed by 9 cm steps. For each case, the percent water volume V_m is measured by the prototype, and the calculate percent volume V_c is calculated through numerical integration as described in Appendix A.

Table 4.1: Outputs of LCD for Varying Inputs Distances given by the Sensor

d	V_c	V_m	ΔV	$abs(\Delta V)$
18	96.30	96.18	0.12	0.12
27	88.77	87.60	1.17	1.17
36	79.37	78.92	0.45	0.45
45	68.83	68.23	0.60	0.60
54	57.63	58.82	-1.19	1.19
63	46.18	47.40	-1.22	1.22
72	34.85	35.20	-0.35	0.35
81	24.04	23.89	0.15	0.15
90	14.20	13.69	0.51	0.51
99	5.94	4.72	1.22	1.22
108	0.43	0.90	-0.47	0.47

From tabulated data, we find that the prototype gives less than 1.3 percent volume error. The mean absolute error (MAE) of the 12 test readings are about 0.7%. It is quiet less than 3% maximum error specified by the sensor data sheets. However, the 3% sensor error was given for an ambient temperature range from 0 to 50 degrees, while the tests were done for only room temperature.

The range of outputs obtained from the sensor is from 10cm to 110cm (the height of the cylinder is 100), because it is sufficient to consider 10cm for the distance of the water surface until the sensor. It can be seen that in D0 and D1 we do not have value for volume because it is out of range. So the ratio of height and volume is fixed because they obey a cylinder dimensions. Thus the sensor gets inputs for calculating the volume till it is calculate for all points.

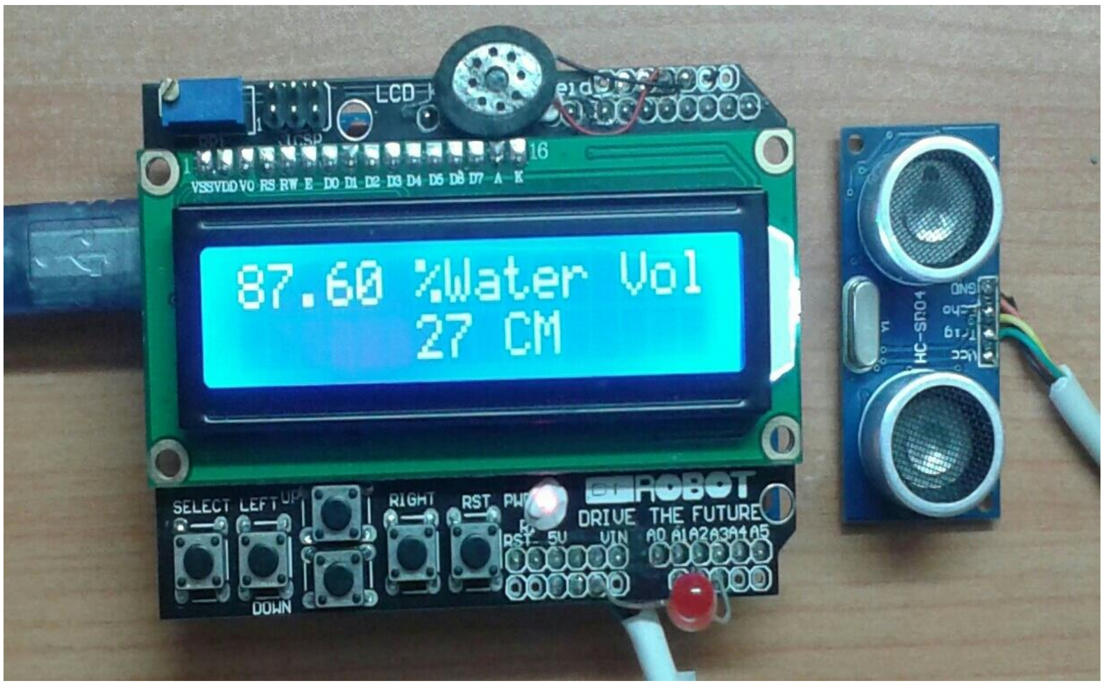


Figure 4.3: A Snapshot of Output Result in d3 = 27 cm

For measuring the water volume in different shapes of water containers we should calculate the height and volume of the tank. As the formulas are given in Appendix A, we can follow the flowchart to get our new look-up table for the new reservoir.

Once we get the look-up table values, we shall enter the look-up table using the buttons of LCD Keypad Shield. In the first step, we should push the Select button to changes the mode from monitor to setup mode. Then we can increments the index of

20-items Look-up-table with the Left button. In the next step, with pushing the up-down buttons, we will increments and decrements the contents of the look-up-table.

Chapter 5

CONCLUSION

In this Thesis, a prototype of a volumetric water measurement instrument has been completed starting from the concepts of system level design. The designed measuring system measures the empty distance from the sensor to the water level in a container, and it calculates the height of the water in the tank. Moreover, it uses a look-up table to calculate the filled percent volume of the water container. The approach of using a look-up table provides an ability to use the prototype for containers of any shape after once calculating and entering the look-up table values to the flash memory of the prototype.

The design is based on an Arduino UNO microcontroller board, which is a common and cheap board with many options of shields, including relay outputs, LCD displays, and much kind of sensor shields. The developed prototype has been constructed mainly from three blocks, an Arduino-UNO board, an LCD with keypad shield, and an ultrasonic sensor. The sensor works at 40 kHz, and measures up to 4 meter distance with maximum 3% error in a temperature range from 0 to 50^{o}C. The 20-point look-up table introduces an additional maximum 1.3% error in calculating the volumetric percentage of the remaining water in the container. However, in the tests carried for 12 water levels, the maximum error is measured only about 1.6%, and the mean absolute error is calculated about 0.7%. With this error figures, the prototype is perfectly suitable for domestic and agricultural usage.

The material cost of the overall prototype is almost 15 dollars, mainly due to the choice of very commonly used building blocks of the design. Additionally, the construction of the prototype is as simple as stacking the blocks on each other.

Finally, the prototype is open to more developments of software to build irritation systems, and pumping systems, along with new alarm conditions. There are extra building blocks to connect the already developed prototype to a network by using Wi-Fi or Bluetooth remote methods for an intelligent house application.

REFERENCES

[1] Banzi, M. (September 2011). Getting Started With Arduino, 2.nd Ed, U.S.A., O'Reilly Books.

[2] Blum, J. (2013). Exploring Arduino Tools and Techniques for Engineering Wizardry, Indiana, J. Wiley & Sons.

[3] (2014). Http://Www.Arduino.Cc.

[4] Huang, S., Http://Arduinodev.Com/Codeblocks.

[5] Muktha Shankari, K., Jyothi, K., Manu, E O., Naveen, I P., Harsha, H. (April 2013). Wireless Automatic Water Level Control Using Radio Frequency Communication, International Journal Of Advanced Research In Electrical, Electronics And Instrumentation Eng., Vol. 2, No. 4, pp. 1320-1324.

[6] Chandra Murmu, I., Kumar Yadav, L. (May 2013). Low Cost Automatic Water Level Control For Domestic Applications, *Thesis*, Electrical Eng. Dept., National Institute of Technology, Rourkela, India.

[7] Monk, S. (2010). 30 Arduino Projects for the Evil Genius, United States, McGraw-Hill Companies.

[8] Patra, A. (May 2011). Development Of Real Time Digital Controller For A Liquid Level System Using Atmega32 Microcontroller, Thesis, Electrical Eng. Dept., National Institute Of Technology, Rourkela, India.

[9] Asran Bin Abdullah, M. (November 2008). Water Level In Tank Using Level Sensor and Pid Controller, Thesis, Electrical Eng. Dept., Univ. Malaysia, Pahang, Malaysia.

[10] Moura, M., Tasa, M., Olejniczak, O., Ahmad, N. (Spring 2012). Level Monitoring System for Waste Oil Containers, School of Eng., Polytechnic Institute of Porto, Portugal, pp. 1-7.

[11] Cytron Technologies Sdn. Bhd. (May 2013). Product User's Manual – HC-SR04 Ultrasonic Sensor, Taman Univ., Johor, Malaysia, pp 1-12.

[12] (2011). Http://www.AccuDIY.com., HC-SR04 Ultrasonic Range Finder.

[13] Kossi, A. (2012). Automated Washing and Level Controlling of a Tank System, Thesis, Technology and Communication Dept., Vassan Ammattikorkeakoulu Univ. applied sciences.

[14] Sunplus Technology co. (August 2003). SPLC780D 16COM/40SEG Controller/Driver, pp. 1-31.

APPENDICES

Appendix A: Calculations of Percent Water Volume in a Cylindrical Container

The fluid volume in a cylinder shaped container may be obtained from the level of the fluid and the size of the container by integrating the horizontal Eros sectional area along the height of fluid. For a vertically aligned cylindrical container the volume comes out proportional to the liquid level because the ones sectional area for all heights is constant. However, the horizontal cross sectional area of a horizontally placed cylinder changes from zero to height × diameter non-proportionally to the height. It is possible to calculate cylindrical vessel volume by analytical methods using trigonometrical geometry and calculus methods. But this Appendix will demonstrate a general numerical method which may be applied for containers of any shape.

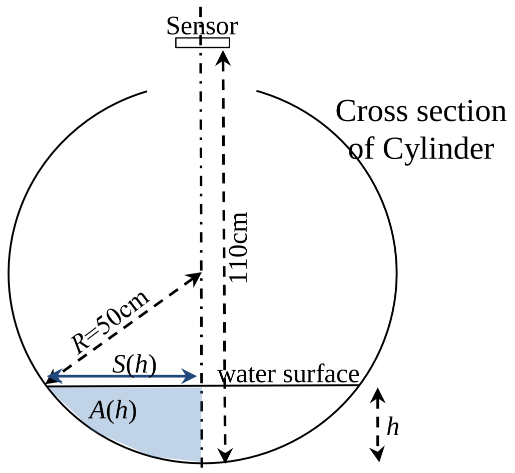


Figure A1.1: Description of Variables Related to the Cylindrical Water Container

The microcontroller needs a look-up table that contains a list of water-levels and their corresponding percent volumes. For the horizontal cylinder the area A under the height h is proportional to the volume, since volume *v*= *l* ×*A*, where the length *l* of the cylinder is constant. Therefore percent cross-sectional area %*A*(*h*) under the height h is always equal to percent volume %V. For the same reason, %V of liquid in the cylinder can be calculated using only %A of the half of the container as shown in Figure 6.1.

Let us divide the height of container to 20 equal length, with steps □h_1= R/20. Starting from bottom, h_0 is zero height, and corresponds to 0% Volume. For the height h_i= i h_1, the surface length S_i is calculated by:

$$S_i = \sqrt{R^2 - (R - i\,h_1)^2}\ ,$$

Also, the total area under the height h_k is:

$$A(h_k) = \sum_{i=1}^{k} h_1\left(S_{i-1} + S_i\right)/\,2$$

Finally, for total n division of the cylinder radius, $A(h_n)$ = $A(R)$ gives the approximation of the half of the cross-sectional area of the cylinder. Therefore, the percent area filled with water is calculated by:

$$\%V(h_k) = \%A(h_k) = \frac{A(h_n) - A(h_k)}{A(h_n)} \times 100$$

The calculations were carried according to the algorithm shown in Figure A1.2.

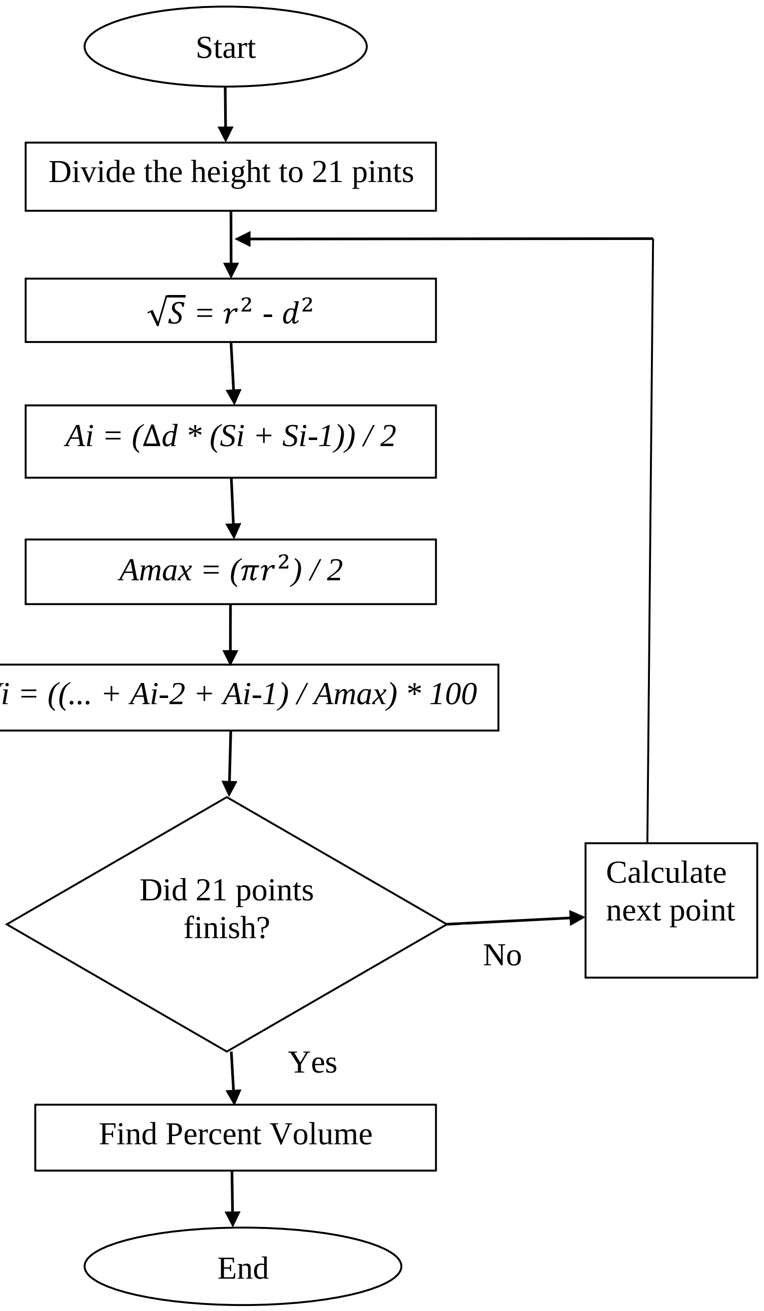


Figure A1.2: Flowchart of Calculating a Water Level in the Cylinder

Table A1.1: Percent Water Volume Calculation
for a Cylindrical Container with 50cm Radius

N	Water height **h_N** (cm)	Distance to Surface **D_N** (cm)	Surface length **S_N** (cm)	Additional Area **ΔA_N** (cm^2)	Cross-sect. Area **A_N** (cm^2)	Percent Volume **%V**
0	0	110	0.0	0.0	0	0.00
1	5	105	21.8	109.0	109	2.81
2	10	100	30.0	150.0	259	6.67
3	15	95	35.7	178.5	438	11.27
4	20	90	40.0	200.0	638	16.43
5	25	85	43.3	216.5	854	22.01
6	30	80	45.8	229.1	1083	27.91
7	35	75	47.7	238.5	1322	34.06
8	40	70	49.0	244.9	1567	40.37
9	45	65	49.7	248.7	1815	46.78
10	50	60	50.0	250.0	2065	53.22
11	55	55	49.7	248.7	2314	59.63
12	60	50	49.0	244.9	2559	65.94
13	65	45	47.7	238.5	2798	72.09
14	70	40	45.8	229.1	3027	77.99
15	75	35	43.3	216.5	3243	83.57
16	80	30	40.0	200.0	3443	88.73
17	85	25	35.7	178.5	3622	93.33
18	90	20	30.0	150.0	3772	97.19
19	95	15	21.8	109.0	3881	100.00
20	100	10	0.0	0.0	3881	100.00

Appendix B: Error Analysis of the Percent Water Volume Calculations

The percent error from the sensor is given as 3% in the data sheet of the sensor. Additional error is expected to stem from the approximated area calculations. The following Excel table contains the calculations for the water filled area A (hk) for a range of divisions n = 2 … 21. From the table we obtain the plot of percent calculation error corresponding to each division value by comparing the total calculated area to the actual area $A_a = (\pi R^2)/2$ of the half circle.

$$\%\text{Error} = \frac{A_a - A(h_n)}{A_a} \times 100$$

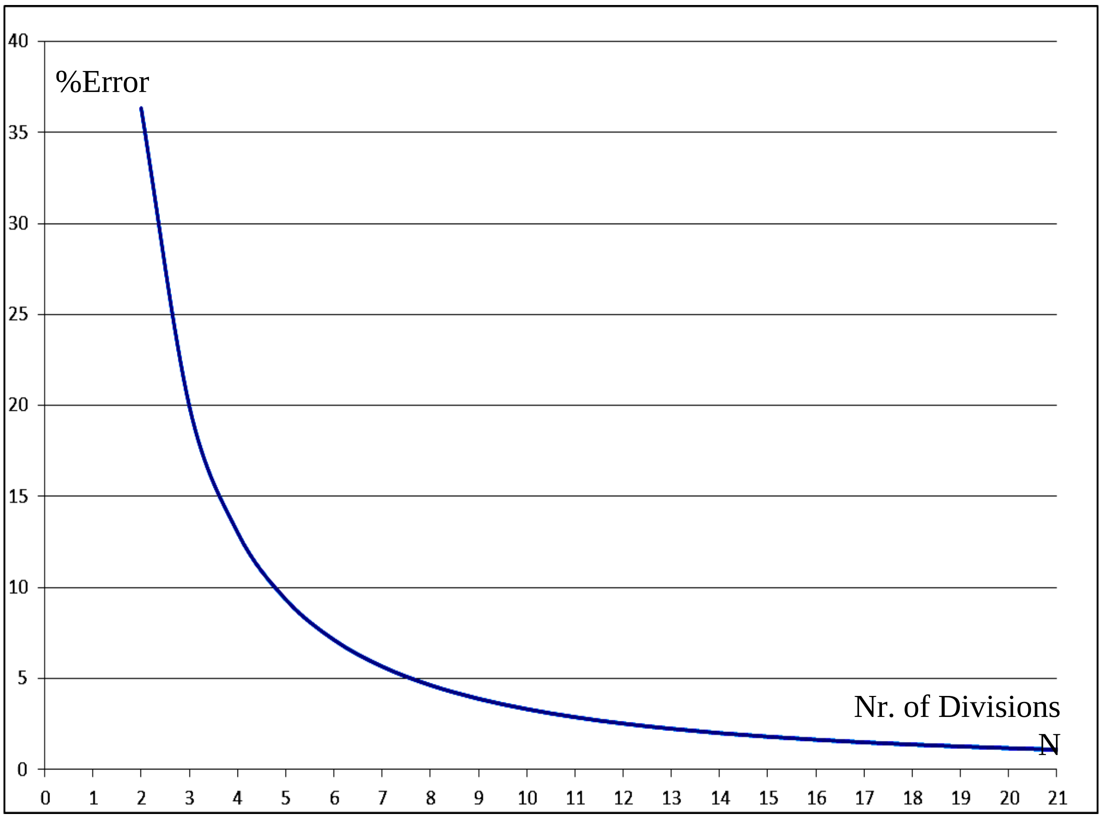


Figure A2.1: Graph of Percent Error E for Number of Divisions N from 2 to 21

Table A2.1: Calculation of Percent Error for Number of Divisions from 2 to 21

	radius=	0.50																			
	n	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00	10.00	11.00	12.00	13.00	14.00	15.00	16.00	17.00	18.00	19.00	20.00	21.00
	h1	0.50	0.33	0.25	0.20	0.17	0.14	0.13	0.11	0.10	0.09	0.08	0.08	0.07	0.07	0.06	0.06	0.06	0.05	0.05	0.05
S	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	1.00	0.50	0.47	0.43	0.40	0.37	0.35	0.33	0.31	0.30	0.29	0.28	0.27	0.26	0.25	0.24	0.24	0.23	0.22	0.22	0.21
	2.00		0.47	0.50	0.49	0.47	0.45	0.43	0.42	0.40	0.39	0.37	0.36	0.35	0.34	0.33	0.32	0.31	0.31	0.30	0.29
	3.00		0.00	0.43	0.49	0.50	0.49	0.48	0.47	0.46	0.45	0.43	0.42	0.41	0.40	0.39	0.38	0.37	0.36	0.36	0.35
	4.00			0.00	0.40	0.47	0.49	0.50	0.50	0.49	0.48	0.47	0.46	0.45	0.44	0.43	0.42	0.42	0.41	0.40	0.39
	5.00				0.00	0.37	0.45	0.48	0.50	0.50	0.50	0.49	0.49	0.48	0.47	0.46	0.46	0.45	0.44	0.43	0.43
	6.00					0.00	0.35	0.43	0.47	0.49	0.50	0.50	0.50	0.49	0.49	0.48	0.48	0.47	0.46	0.46	0.45
	7.00						0.00	0.33	0.42	0.46	0.48	0.49	0.50	0.50	0.50	0.50	0.49	0.49	0.48	0.48	0.47
	8.00							0.00	0.31	0.40	0.45	0.47	0.49	0.49	0.50	0.50	0.50	0.50	0.49	0.49	0.49
	9.00								0.00	0.30	0.39	0.43	0.46	0.48	0.49	0.50	0.50	0.50	0.50	0.50	0.49
	10.00									0.00	0.29	0.37	0.42	0.45	0.47	0.48	0.49	0.50	0.50	0.50	0.50
	11.00										0.00	0.28	0.36	0.41	0.44	0.46	0.48	0.49	0.49	0.50	0.50
	12.00											0.00	0.27	0.35	0.40	0.43	0.46	0.47	0.48	0.49	0.49
	13.00												0.00	0.26	0.34	0.39	0.42	0.45	0.46	0.48	0.49
	14.00													0.00	0.25	0.33	0.38	0.42	0.44	0.46	0.47
	15.00														0.00	0.24	0.32	0.37	0.41	0.43	0.45
	16.00															0.00	0.24	0.31	0.36	0.40	0.43
	17.00																0.00	0.23	0.31	0.36	0.39
	18.00																	0.00	0.22	0.30	0.35
	19.00																		0.00	0.22	0.29
	20.00																			0.00	0.21
	21.00																				0.00
A	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	1.00	0.13	0.08	0.05	0.04	0.03	0.02	0.02	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	2.00	0.25	0.24	0.17	0.13	0.10	0.08	0.07	0.06	0.05	0.04	0.04	0.03	0.03	0.03	0.03	0.02	0.02	0.02	0.02	0.02
	3.00		0.31	0.29	0.23	0.18	0.15	0.13	0.11	0.09	0.08	0.07	0.06	0.06	0.05	0.05	0.04	0.04	0.04	0.03	0.03
	4.00			0.34	0.32	0.26	0.22	0.19	0.16	0.14	0.12	0.11	0.10	0.09	0.08	0.07	0.07	0.06	0.06	0.05	0.05
	5.00				0.36	0.33	0.29	0.25	0.22	0.19	0.17	0.15	0.13	0.12	0.11	0.10	0.09	0.09	0.08	0.07	0.07
	6.00					0.36	0.35	0.31	0.27	0.24	0.21	0.19	0.17	0.16	0.14	0.13	0.12	0.11	0.10	0.10	0.09
	7.00						0.37	0.35	0.32	0.29	0.26	0.23	0.21	0.19	0.18	0.16	0.15	0.14	0.13	0.12	0.11
	8.00							0.37	0.36	0.33	0.30	0.27	0.25	0.23	0.21	0.19	0.18	0.17	0.15	0.14	0.14
	9.00								0.38	0.36	0.34	0.31	0.29	0.26	0.24	0.22	0.21	0.19	0.18	0.17	0.16
	10.00									0.38	0.37	0.34	0.32	0.30	0.27	0.25	0.24	0.22	0.21	0.19	0.18
	11.00										0.38	0.37	0.35	0.33	0.30	0.28	0.27	0.25	0.23	0.22	0.21
	12.00											0.38	0.37	0.35	0.33	0.31	0.29	0.28	0.26	0.24	0.23
	13.00												0.38	0.38	0.36	0.34	0.32	0.30	0.28	0.27	0.25
	14.00													0.38	0.38	0.36	0.34	0.32	0.31	0.29	0.28
	15.00														0.39	0.38	0.36	0.35	0.33	0.31	0.30
	16.00															0.39	0.38	0.37	0.35	0.33	0.32
	17.00																0.39	0.38	0.37	0.35	0.34
	18.00																	0.39	0.38	0.37	0.36
	19.00																		0.39	0.38	0.37
	20.00																			0.39	0.38
	21.00																				0.39
	Aa	.393	.393	.393	.393	.393	0.393	.393	.393	.393	.393	.393	.393	.393	.393	.393	.393	.393	.393	.393	.393
	Ac	.250	.314	.342	.356	.365	0.370	.374	.377	.380	.381	.383	.384	.385	.386	.386	.387	.387	.388	.388	.388
	Error%	36.3	19.9	13.0	9.35	7.13	5.667	4.64	3.89	3.32	2.88	2.53	2.24	2.01	1.81	1.65	1.50	1.38	1.27	1.18	1.09

According to the error analysis, having 20 linear divisions (21 points) brings maximum 1% error additional to the 3% error of the sensor reading, and makes the total error 4%.

Appendix C: Arduino Source Code

```
#include <LiquidCrystal.h>
LiquidCrystallcd(8, 9, 4, 5, 6, 7); // the pins used on the LCD panel
intlcd_key = 0; // values used by the panel and buttons
intadc_key_in = 0;
inti=0;
float vh=0; //upper bound HV
float vl=0; //lower bound LV
int dl=0;
int dh=0;
float cv=0; //current volume
int x=1;
float y=0;
int no=0;
int mode=0;
int dis[22]={0, 100, 95, 90, 85, 80, 75, 70, 65, 60, 55, 50, 45, 40,
35, 30, 25, 20, 15, 10, 5, 0};
float vol[22]={0, 0, 1.38, 4.68, 8.87, 13.69, 19, 24.67, 30.63,
36.79, 43.08, 50, 55.78, 62.07, 68.23, 74.19, 79.86, 85.17, 89.99,
94.18, 97.48, 100};
char* mstring[]= {"Ch D", "Ch V", "RUN"};
#define btnRIGHT  0 // read the buttons
#define btnUP     1
#define btnDOWN   2
#define btnLEFT   3
#define btnSELECT 4
#define btnNONE   5
#define CM 1//Centimeter
#define INC 0 //Inch
#define TP A1 //Trig_pin
#define EP A2 //Echo_pin
#define led A3
#define buzzer1 3
int sound = 300;
intread_LCD_buttons()
{
adc_key_in = analogRead(0); // read the value from the sensor
// my buttons when read are centered at these valies: 50, 195, 380,
555, 790
if (adc_key_in> 1000) return btnNONE; // I make this the 1st option
for speed reasons since it will be the most likely result
if (adc_key_in< 50)   return btnRIGHT; // For V1.0 comment threshold
if (adc_key_in< 195)  return btnUP;
if (adc_key_in< 380)  return btnDOWN;
if (adc_key_in< 555)  return btnLEFT;
if (adc_key_in< 790)  return btnSELECT;
}
void setup()
{
lcd.begin(16,2);
Serial.begin(9600); // 9600 bps
pinMode(TP, OUTPUT); // set TP output for trigger
pinMode(EP, INPUT);// set EP input for echo
pinMode(led, OUTPUT);
pinMode(buzzer1, OUTPUT);
}
void loop()
{
lcd_key = read_LCD_buttons(); // read the buttons
switch (lcd_key)              // depending on which button was
pushed, we perform an action
{
case btnRIGHT:
{
if (mode==2)
{
}
break;
}
case btnLEFT:
{
if (mode==0)
{
if (no>20)
{
no=0;
}
no=no+ 1;
delay(200);
lcd.clear();
lcd.setCursor(4,1);
lcd.print(no-1);
lcd.setCursor(7,1);
lcd.print(dis[no]);
i=0;
if (no>20)
{
no=0;
}
}
if (mode==1)
{
if (no>20)
{
no=0;
}
no=no+ 1;
delay(200);
lcd.clear();
lcd.setCursor(4,1);
lcd.print(no-1);
lcd.setCursor(7,1);
lcd.print(vol[no]);
lcd.setCursor(11,1);
lcd.print("%");
i=0;
if (no>20)
{
no=0;
}
}
break;
}
case btnUP:
{
if (mode==0)
{
x=0;
x=dis[no];
x=x+ 1;
lcd.setCursor(10, 1);
dis[no]=x;
lcd.print(x);
delay(200);
}
if (mode==1)
{
y=0;
y=vol[no];
y=y+ 0.01;
lcd.setCursor(10, 0);
vol[no]=y;
lcd.print(y);
delay(200);
}
break;
}
case btnDOWN:
{
if (mode==0)
{
x=0;
x=dis[no];
x=x- 1;
lcd.setCursor(10, 1);
dis[no]=x;
lcd.print(x);
delay(200);
}
if (mode==1)
{
y=0;
y=vol[no];
y=y- 0.01;
lcd.setCursor(10, 0);
vol[no]=y;
lcd.print(y);
delay(200);
}
break;
}
```

```
case btnSELECT:
{
delay(500);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Select M ");
lcd.setCursor(0,1); // move to the begining of the second line
mode=mode+1;
if (mode>2)
{
mode=0;
}
lcd.print(mstring[mode]);
break;
}
case btnNONE:
{
if (mode== 2)
{
long microseconds = TP_init();
long distance_cm = Distance(microseconds);
lcd.clear();
lcd.setCursor(6, 1);
lcd.print(distance_cm);
lcd.print(" CM ");
distance_cm=distance_cm-10;
dh=dis[22-(distance_cm/5)];
dl=dis[22-(distance_cm/5)-1];
vh=vol[22-(distance_cm/5)];
vl=vol[22-(distance_cm/5)-1];
distance_cm=distance_cm;
cv=(((dh-dl)/(vh-vl))*(distance_cm-dl))+vl;
lcd.setCursor(0, 0);
lcd.print(cv);
lcd.setCursor(6, 0);
lcd.print("%Water Vol");
if (distance_cm>= 85)
{
digitalWrite(led, HIGH);
tone(buzzer1, 2);
}
else
{
digitalWrite(led, LOW);
noTone(buzzer1);
}
}
break;
}
}
}
long Distance(long time)
{
long distacne;
```

```
distacne = time / 29 / 2; // Distance_CM  = ((Duration of high
level)*(Sonic :340m/s))/2
// = ((Duration of high level)*(Sonic : 0.034cm/us))/2
// = ((Duration of high level)/(Sonic : 29.4cm/us))/2
return distacne;
}
long TP_init()
{
digitalWrite(TP, LOW);
delayMicroseconds(2);
digitalWrite(TP, HIGH); // pull the Trig pin to high level for more
than 10us impulse
delayMicroseconds(10);
digitalWrite(TP, LOW);
long microseconds = pulseIn(EP, HIGH); // waits for the pin to go
HIGH, and returns the length of the pulse in microseconds
return microseconds; // return microseconds
}
```