

A
PROJECT REPORT
on

**ULTRASONIC BASED MEASUREMENT SYSTEM FOR
LIQUID LEVEL AND TYPE MONITORING**
Submitted in partial fulfillment of the requirements for the award of the degree of
BACHELOR OF TECHNOLOGY

in
ELECTRONICS AND COMMUNICATION ENGINEERING

By

M.G. DEEPAK SAI KUMAR (19JD1A0440)

A.VAMSI KRISHNA (19JD1A0401)

B.B.S.KRISHNA VAMSI (20JD5A0401)

Under the esteemed guidance of

P. SURYANARAYANA M.Tech.

Assistant Professor

Department of ECE



**DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING
ELURU COLLEGE OF ENGINEERING AND TECHNOLOGY
DUGGIRALA (V), PEDAVEGI (M), ELURU-534004
APPROVED BY AICTE-NEW DELHI&AFFILIATED TO JNTUK-KAKINADA**

2019 -2023

ELURU COLLEGE OF ENGINEERING & TECHNOLOGY

(Affiliated to JNTUK-KAKINADA, Approved by AICTE-NEW DELHI)

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING



CERTIFICATE

This is to certify that the Project Report entitled **“ULTRASONIC BASED MEASUREMENT SYSTEM FOR LIQUID LEVEL AND TYPE MONITORING”** being submitted in partial Fulfillment for the award of the degree **Bachelor of Technology** in **Department of Electronics & Communication Engineering** to the **Jawaharlal Nehru Technological University, Kakinada** is a record of bona-fide work carried out by

M.G.DEEPAK SAI KUMAR (19JD1A0440)

A.VAMSI KRISHNA (19JD1A0401)

B.B.S.KRISHNA VAMSI (20JD5A0401)

under my guidance and supervision.

PROJECT GUIDE

P. SURYANARYANA M. Tech

Assistant Professor

Department of ECE

HEAD OF THE DEPARTMENT

Dr. B. RAJA RAO M.Tech.,Ph.D

Professor & HOD

Department of ECE

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

In the beginning we want to elucidate that this project would have been a distant dream without the grace of Almighty God who has blessed us with a drop of knowledge from his mighty ocean.

We are very grateful to my project guide **P. Suryanarayana** M.Tech for his inspiration, guidance, constant supervision and constructive criticism in successful completion of this project. he provided organization, supported enthusiastic discussions, in depth reviews and valuable references.

We are very grateful to the Head of the ECE Department **Dr. B. RAJARAO** Ph.D Professor for his valuable guidance, motivation and endless supply of new ideas and technical support for this project.

We are thankful to our **Principal Dr. P. BALAKRISHNA PRASAD** Ph.D for permitting and encouraging me in doing this project.

We extend our sincere thanks to **Sri V. Raghavendra Rao**, Chairman of our college for providing sufficient infrastructure and good environment in the college to complete my course.

Great acknowledgement is expressed to **Coordinator, Teaching and Non-Teaching Staff Members** whose guidance cannot be ignored in completing this project in time

M.G.Deepak Sai Kumar (19JD1A0440)

A.Vamsi Krishna (19JD1A0401)

B.B.S. Krishna Vamsi (20JD5A0401)

DECLARATION

We here by declaring that the project work entitled “**Ultrasonic based Measurement System for Liquid Level And Type Monitoring**” submitted to JNTU Kakinada, is a record of original work done by us. This project work is submitted in the partial fulfillment for the degree of Bachelor of Technology in Electronics and communication engineering. The results embedded in this thesis have not been submitted to any other University or Institute for the award of any degree or Diploma.

M.G.Deepak Sai Kumar (19JD1A0440)

A.Vamsi krishna (19JD1A0401)

B.B.S. Krishna Vamsi (20JD5A0401)

CONTENTS

CHAPTER NO.	NAME	PAGE NO
	LIST OF FIGURES	i
	LIST OF TABLES	iii
	ABSTRACT	iv
1	INTRODUCTION	1
	Objective	2
2	LITERATURE SURVEY	4
	2.1 Wearable based FMS	6
	2.2 Sensor instrumented Smart-bottles	6
3	EXISTING SYSTEM	8
	3.0 Methodology	8
	3.1 Components used	9
	3.2 System implementation	10
	3.3 Testing and evaluation	11
	3.4 Implementation	12
	3.5 Discussions	13
	3.6 Conclusions and Futurescope	13
4	PROPOSED SYSTEM	15
	4.0 Design Considerations	15
	4.1 Hardware platform	16
	4.2 Method of operation	17
	4.2.1 Design Flow	17
	4.2.2 Bottle-state Recognition	20
	4.2.3 Volume estimation	21
	4.2.4 Temperature compensation	23
	4.2.5 Type Detection	24
	4.2.6 Performance Metrics	24
	4.3 Data collection	28
	4.4 Volume Estimation	29
	4.4.1 Regression model comparison	29
	4.4.2 Leave one volume out (LOVO)	30
5	COMPONENTS USED	31
	5.1 Ultrasonic sensor	31

	5.1.1 Variation of speed of sound	34
	5.2 RGB color sensor	35
	5.3 Temperature sensor	37
	5.4 Accelerometer	40
	5.5 Bluetooth Module	42
	5.6 SD card Breakout Board	44
6	EMBEDDED SYSTEM	48
	6.0 Introduction to Embedded systems	48
	6.1 Examples of Embedded systems	49
	6.2 How does an Embedded system work?	49
	6.3 Characteristics of Embedded systems	50
	6.4 Structure of Embedded systems	50
7	SOFTWARE DESCRIPTION	52
	7.0 Introduction to software modules	52
	7.1 Arduino IDE	52
	7.1.1 Arduino IDE initial setup	53
	7.1.2 IDE: Board Setup	54
	7.1.3 IDE: COM port setup	54
	7.1.4 Testing your settings	55
	7.2 Burning bootloader of Arduino	55
8	RESULTS AND CONCLUSION	60
9	FUTURE SCOPE	62
	9.1 Future scope	62
	9.2 Extension of this application	63
10	ARDUINO SKETCH	65
11	REFERENCES	69

LIST OF FIGURES

FIGURE NO.	NAME OF THE FIGURE	PAGE NO.
3.1	Basic block construction of existing system	9
3.2	The physical connections of Existing system	11
3.3	Snapshot of experiment conducted on a water tank to measure water level using the UMS module	12
4.1	Electronic components of proposed system	16
4.2	Operational state machine during both calibration phase and regular operation	18
4.3	Signal processing and machine learning flow of the algorithms in the system for bottle-state recognition	20
4.4	Machine learning flow for volume estimation	21
4.5	Volume measurement procedures	22
4.6	Color detection with RGB color sensor	23
4.7	Proposed neural network for color embedding	27
4.8	Current LIDS prototype	28
5.1	Operation of Ultrasonic sensor	31
5.2	Waterproof Ultrasonic sensor A02YYUW	34
5.3	The graph to show variation of speed of sound with respect to temperature	34
5.4	Reflected Light for different object colors	35
5.5	TCS3200 RGB sensor	36
5.6	DSB1820 Waterproof temperature sensor	38
5.7	ADXL335 Accelerometer	40
5.8	Accelerometer sensor MEM mechanism	41
5.9	Bluetooth Module HC-05	42
5.10	Pin description of HC-05 Bluetooth module	43
5.11	SD card Breakout Board	44
5.12	Esp8266-01 WiFi module	47
6.1	Arduino microcontroller connected with wires	48
6.2	Structure of Embedded system	51
7.1	Tool bar in Arduino IDE	53
7.2	Arduino IDE window	54

7.3	Connections for burning the bootloader of Arduino Nano using Arduino Uno	56
7.4	Connections for burning the bootloader, Using ICSP Header	56
7.5	Selecting the programmer options	57
7.6	Burning the bootloader	58
7.7	Acknowledgment after successful burning	59

LIST OF TABLES

3.1	Variation in temperature and relative humidity at three different positions for different storage tanks	10
8.1	Accuracy based on different Illuminance conditions	49
8.2	Relative absolute error approximated based on inputs	49

ABSTRACT

Low-cost ultrasonic sensors are widely used for non-contact distance measurement problems. Speed of ultrasonic waves is greatly affected by environmental conditions such as temperature and relative humidity among a few other parameters. Presence of acoustic and electronic noise also influence ultrasonic sensor-based distance measurement system. Existing standard techniques assume that the temperature and relative humidity level remains constant throughout the measurement medium. In our proposed system, we measure water level in storage containers, which exhibits a gradient of temperature and relative humidity across the measurement medium. Hence, the standard Ultrasonic Measurement System (UMS) is not able to estimate distance accurately. Fluid intake tracking is crucial in providing interventions that assist individuals to stay hydrated by maintaining an adequate amount of fluid. It also helps to manage calorie intake by accounting for the amount of calorie consumed from beverages. While staying hydrated and controlling calorie intake is critical in both physical wellness and cognitive health, existing technologies do not provide a solution for monitoring both fluid type and fluid volume.

To address this limitation, we present the design, implementation, and validation of Liquid Intake Detection System for real-time tracking of fluid intake type and volume. The system devises a sensing module that is composed of ultrasonic, RGB color, temperature, and accelerometer sensors as well as a computational framework for fluid intake type classification, volume estimation, and bottle-state-recognition. And the system uploads data to a web page where we can monitor the data up-to a long duration of time. We conduct extensive experiments to collect data in a variety of bottles and environmental settings.

Keywords – *Bottle, Smart sensors, liquid intake, ultrasonic sensor, ultrasonic measurement system (UMS), distance measurement, temperature, relative humidity, color.*

CHAPTER 1

INTRODUCTION

Maintaining an adequate amount of fluid intake is critical in both physical and cognitive wellness and health. Prior research emphasizes the serious physical and cognitive consequences of dehydration, as well as the preventive care benefits of taking a sufficient amount of fluid. For example, while dehydration could result in complications in the cardiovascular system, thermoregulation, metabolism, and central nervous functions, sufficient fluid intake helps reduce the incidence of bladder and colon cancer. Dehydration can also result in negative cognitive performance, such as poor concentration, increased reaction time, degraded visual attentiveness, short-term memory problems, moodiness, and anxiety. Recent research suggests that even a mild degree of dehydration (e.g., a body water loss of 1%–2%), which may occasionally occur throughout daily routines, can impair cognitive abilities.

Most of the fluid intake (i.e., approximately 81%) should be achieved through fluids rather than solid foods. A major challenge in maintaining a sufficient fluid level and preventing dehydration is that people are often unaware of their hydration status. A recent survey reported that, although most people understand that sufficient water consumption is important, approximately 75% of the American population falls short of the recommended daily fluid intake (i.e., 2.7 L for women and 3.7 L for men). In addition to gender, healthcare providers recommend different amounts of daily fluid intake based on age, health condition, and one's activity level. For example, it is recommended that young children consume fluid regularly to improve memory recall, and patients with cancer consume fluid frequently to improve their health condition because they often do not feel thirsty even when dehydrated.

Many mobile systems (e.g., smart bottles) have been introduced to 1) serve as a reminder for the users to adhere to maintaining an adequate hydration level and 2) track the volume of fluid intake using various types of sensors and analytic methods. There exist many commercially available products, such as HydraCoach R, Hidrate Spark R, and H2OPal R, which can track fluid intake. These smart bottles estimate the amount of fluid intake using sensors such as resistive sensor and flow meter. Users can track their fluid intake via mobile applications that these products provide and they can also use these systems to set daily hydration goals. More recently, researchers presented alternative technologies for measuring fluid intake and estimating body hydration. FluidMeter, AutoHydrate and are examples of such studies that use wearable systems to estimate fluid intake. In addition, studies such as

and use a bottle-attachable Inertial Measurement Unit (IMU) sensor for volume estimation purposes. However, the above-mentioned technologies mainly focus on tracking the volume of fluid intake and therefore fall short in identifying the type of consumed fluids.

Objective

To develop a viable solution for ubiquitous monitoring of fluid intake, it is essential to monitor the type of consumed fluid, because different fluids have different hydrating capabilities depending on their electrolyte and glucose contents. In other words, the suggested amount of fluid intake may vary depending on the beverage type. Furthermore, identification of the fluid type could be effective in monitoring the user's cheating behaviors on the prescribed/recommended regimen (e.g., drinking sugar-sweetened high-calorie beverages) or tracking one's calorie intake from fluid consumption. For instance, prior research reported that individuals consume, on average, 10%–15% of their total daily calories from sugar sweetened beverages. Hence, the development of a personalized tracking solution for fluid intake—in terms of both the volume and the type is in great need to assist users to achieve their hydration goals in everyday living situations.

In this, we introduce a Liquid Intake Detection System a mobile sensor system attached to the lid of a reusable water bottle (e.g., a tumbler) from inside, capable of automatically monitoring fluid intake volume, type, and temperature. The proposed system also detects important activities related to fluid intake, such as drinking from the bottle and opening the lid, which enables the opportunistic but continuous monitoring of the target fluid volume and type, further optimizing the power consumption of the embedded system. LIDS consists of several sensor modalities including an ultrasonic sensor, an RGB (Red, Green, and Blue) colour sensor, a temperature sensor, and an accelerometer. These sensors are coupled with an ultra-low power micro-controller and a wireless module for real-time sensor data sampling, processing, and transmission.

The system leverages machine learning algorithms to estimate and recognize the volume and the type of the fluid content in the bottle. We perform a series of comprehensive analyses on the data collected in different experimental settings to assess accuracy, robustness, and generalizability of our system for its use in real-world environments. Moreover, we propose a novel feature embedding to tackle a significant challenge of deploying our system in the real world which is predicting fluids in new bottles that have not been in the training data. Our results demonstrate an accuracy of 97.6% for detecting fluid type when training data contain data of the bottle that is being tested and root relative squared

error of 1.1% for estimating volume by leveraging our proposed feature embedding classification method and Random Forest regression algorithm, respectively.

The feature embedding methods achieved an average accuracy of 84.8% while evaluating with unseen bottles.

CHAPTER-2

LITERATURE SURVEY

This chapter reviews literature related to commonly used liquid level measurement techniques and highlights existing works related to ultrasonic level measurements.

There are various types of level measurement sensors are used to measure liquid level [2], [13]. These are mainly float type [14], [15], capacitive type [2], [16], [17], optical type [18], [19], radar type [2], [20] and ultrasonic type [2]–[5], [11], [12], [21], [22]. A float type level switch consists of a magnetic float which floats on the liquid surface. As the fluid level rises, the float moves vertically and this motion is used to measure the level. Float level devices are contact-type point level sensors, which have been used since early times due to their simple structure and reasonable cost. However, float level sensors suffer from low accuracy and frequent maintenance. Capacitive level sensors are made from two copper plates and the dielectric constant observed between two plates is proportional to the water level [16]. The dielectric constant of the liquid must be known for this type of measurement. The main advantages of these capacitive level measurement sensors include broad application range and good accuracy [17]. Performance of capacitive sensors gets affected by the change in the dielectric constant which varies with temperature of the liquid to be measured. These sensors are well-suited for both point and continuous level measurements.

In optical type level measurement, the reflective property of light is exploited for the measurement of liquid level. Optical level sensor consists of an infrared light-emitting diode (LED) and a light receiver. Light from the LED is directed to a prism and is reflected from the prism to the receiver when there is no liquid. When the sensor is immersed in liquid, the light is refracted out into the liquid, leaving little or no light to reach the receiver. The amount of received light by the receiver transistor indicates the liquid level. It is also a contact-type and point level detection sensor. The optical type requires frequent maintenance and it is adversely affected by the change in the reflective property of the medium [19]. Unlike ultrasonic sensors which use sound waves, radar level sensors use radio waves [2]. Main advantages of radar level sensors are high accuracy, non-contact type, and continuous level measurement. However, radar level sensors are of high cost and the cost increases exponentially with an increase in desired accuracy. Ultrasonic sensors are low-cost, long-range, non-contact and continuous level measurement devices. These sensors are widely used in many applications because of their simplicity of use, high level of safety and measurement resolution, ease of installation and very little maintenance.

Carullo et al. [3] described an ultrasonic distance measurement technique for automotive applications to measure the height from the ground to a vehicle body. In their experiment the measured distance is in the range of 100–600 mm and the temperature is in the range of 0–40 °C. The standard distance measurement uncertainty reported in their experiment is 1 mm. In [17], authors presented an extensive review of the existing state-of-the-art techniques for liquid level monitoring along with a comparison between capacitive and ultrasonic water level measurement system. Terzic et al. [11] developed Support Vector Machine (SVM) based signal processing and classification approach coupled with a single ultrasonic sensor to accurately determine the fuel level in an automotive fuel tank under dynamic conditions. Bucci and Landi [22] presented a novel algorithm for the measurement of signal transit time and applied it to the ultrasonic sensor for water level measurement in a water tank. The results obtained from their experiment indicated a mean error of 0.5 mm for water level ranging from 100 to 1000 mm in ambient temperature conditions.

In [10], Canali et al. designed an ultrasonic measurement system that can operate in air medium with temperature range from –20 °C to +100 °C and measure distance up to 100 cm. They only considered the effect of temperature on the speed of ultrasonic waves. Matsuya et al. [23] proposed a new method for liquid-level measurement utilizing wedge waves (generated by the ultrasonic transducer) and then demonstrated through finite element method (FEM) simulation using both interface echo method and end echo method. The standard deviations and the uncertainties of their measurement method observed to be 0.65 mm and 0.21 mm respectively for interface echo method, and 0.39 mm and 0.12 mm respectively for end echo method. Zhang et al. [24] proposed a novel method for ultrasonic sensor measurement based on balanced echo energy for level measurement. This method uses the balance of ultrasonic echo energy received by two sensors to determine the liquid level from outside the sealed container.

In [12], the authors developed a sensing device that can monitor flash flood and traffic congestion in urban cities. This device consists of an ultrasonic sensor and a passive infrared temperature sensor. Machine learning techniques such as ANN, fuzzy logic and nonlinear regression were used to predict the water level. In their experiment, raw distance measurement varies by 12 cm but with their proposed method the estimated distance error reduced to less than 2 cm. Carullo et al. described a performance improvement technique for ultrasonic distance sensors using two-level neural networks. Neural network is used to process the ultrasonic echo's to improve measurement accuracy. Their method could limit the

error to 0.5 mm over a distance range up to 500 mm. All ultrasonic liquid level sensing techniques discussed above have different application areas, different measurement ranges, and varied ranges of temperatures. None of the aforementioned methods explicitly considered the effect of relative humidity and gradient of temperature and humidity in the measurement medium. Therefore, in this work, we propose an ANN based adaptive UMS to measure the distance with higher accuracy wherein, the error is limited to millimeter range.

2.1 Wearable-based Fluid Monitoring System:

Several studies have introduced wearable-based systems to estimate fluid intake volume. Fluid Meter is a ubiquitous system that can track the amount of fluid intake by using sensors embedded in smartwatches. Fluid Meter estimates the overall amount of fluid intake in grams with an estimation error of 15% and recognizes drinking gestures with an accuracy of 80.8%. Mengistu et al. presented Auto Hydrate, a wearable hydration monitoring system consisted of a microphone attached to the neck to monitor acoustic signals related to drinking and a smartwatch for collecting body activities and gestures. Based on an experiment involving eight individuals, authors reported a drinking detection accuracy of 91.5% and a body activity classification accuracy of 89.1%. Wearable-based solutions are less-obtrusive and less-invasive. However, these systems often suffer from low estimation accuracy for the amount of consumed fluid due to an obvious reason that the estimation relies on the counts of drinking body gestures. Furthermore, these wearable-based solutions cannot effectively monitor fluid intake when the user takes gestures other than the conventional drinking gestures (e.g., using a straw to drink from a bottle).

2.2 Sensor-instrumented Smart-Bottles:

One of the earliest sensor-equipped smart-bottles that could track and display daily fluid intake of the user includes HydraCoach R. The device uses an oral-suction-activated flow meter to measure the volume of the liquid moving from the bottle through a tube. Chiu et al. proposed Playful Bottle that measures fluid intake volume using phone cameras. The average error rate obtained from 16 participants was 3.86%.

Hidrate Spark R is a commercial product that tracks the hydration level using accelerometer and touch sensors. The smartphone application associated with the device stores the hydration-related data along with the location information where drinking events occur. Borofsky et al. validated the fluid intake measurement of Hidrate Spark R and showed an error rate of 3% compared to the user-reported measurement. In addition to these popular reusable water-bottles that integrate embedded sensors, there exist other smart bottles such as

Moikit R, Thermos R, and H2OPal R that have been used to measure fluid intake and temperature. Moikit R monitors water volume and the amount of fluid consumption by measuring air pressure in the bottle, which yields an accuracy of 5%. Thermos R has a sensor tube that measures the liquid volume with a resistive sensor. Jovanov et al. presents intelligent water bottle that can measure the amount of liquid in a bottle, monitor activity using inertial sensors, and physiological parameters using a touch and photoplethysmographic sensor. Kreutzer et al. introduces a fluid monitoring method that consists of passive sensor cups including a resistive sensor, an RFID transponder, and an RFID reader. H2OPal R uses an accelerometer and a weight sensor to monitor the water level in the bottle. Dong et al. proposed a miniaturized embedded system equipped with just an accelerometer that can be attached to a regular bottle to monitor fluid intake. The system captures and detects acceleration signatures related to drinking events, such as pouring and drinking. Then it classifies the amount of consumed liquid based on drinking motion which is captured by accelerometer. The system transmits information regarding the detected events to a smartphone via Bluetooth. The authors show that the system can achieve 99% accuracy in detecting drinking events and 75% accuracy in estimating the volume of fluid intake. Liu et al. uses a 3D printed smart cup attached with a single accelerometer which can detect drinking events and recognize complete periods of drinking. Soubam et al. proposed a sensor-instrumented base that can be attached to a water bottle.

The system can also be linked to a smartwatch to detect drinking motion with an accuracy of 93.53%. Although the above-mentioned prior research has been successful in detecting fluid intake using various types of sensors embedded in water bottles, most of the systems focus on estimating the volume of consumed fluid. There is a gap in our knowledge on how to develop technologies that can monitor both the type and volume of the fluid consumed by the user in a minimally-invasive manner. The proposed technology in this paper, LIDS, bridges the gap by introducing an integrated system that can monitor the type and volume of the liquid inside a bottle based on an array of sensors, including accelerometer, ultrasonic distance, and RGB color sensors. Furthermore, LIDS also monitors activities, such as whether a user drinks from the bottle or opens the lid, or whether the bottle is positioned stationary or moving (e.g., walking), which are used to opportunistically sample sensor data and further optimize the power efficiency of the system. The proposed system is complemented by utilizing machine learning algorithms for detecting drinking activities, classifying the type of liquid content, and estimating its volume.

CHAPTER 3

EXISTING SYSTEM

3.0 Methodology:

Ultrasonic rangefinder depends on the Time-of-Flight (ToF) of the signal to measure distance. The distance D is calculated from the ToF using the equation, $D = (\text{ToF} * c)/2$, where c is the speed of sound (m/s). The speed of sound in dry air at Standard Temperature and Pressure (STP) is $331.45 \text{ m/s} \pm 0.05$. (STP: 273.15 K , $1.01325 \times 10^5 \text{ Pa} = 1 \text{ atm}$)

Most important environmental parameters that affect the speed of ultrasonic sound waves in air are temperature, relative humidity and to a lesser extent other gases present in the medium. Acoustic noise also has a major effect on ultrasonic sensor operation and distance measurements. Both environmental parameters and acoustic noise induce uncertainty in ultrasonic sensor-based distance measurements. Nowadays, water is scarce and a more valuable resource as the gap between demand and supply increases day by day. Thus, proper management of water resources using required technology is the need of the hour. Storage tanks are used to store bulk water and monitoring water level in storage tanks is very important for efficient water management. The water level can be determined using the following equation $L = H - D$ (1) where L is the water level, H is the height of the tank measured from ultrasonic sensor position to zero level, $D = (\text{ToF} * c)/2$, represents the distance from measuring device (ultrasonic sensor) to water level or surface.

Speed of sound in a medium increase as the temperature of the medium increases and it will lead to incorrect distance or level measurements [8]. The effect of temperature and relative humidity can be compensated using observations of temperature and humidity sensor along with the ultrasonic sensor. Ultrasonic level measurement system will yield noticeable accuracy if temperature and relative humidity remain constant between sensor position and the water level. Despite this, there still exists error in the measurement which is significant for many level measurement applications. This error is due to uncertain parameters like electronic noise and the variation (gradient) of temperature and relative humidity throughout the ultrasonic signal travel path between sensor position and water level. Especially, when the water tank diameter is larger, a small variation in level measurement will result in erroneous volume measurement of water contained in the tank. When the tank is exposed to sun, the temperature and relative humidity gradient induces significant error in level measurement.

Most of the existing works done so far in this field focused on compensating the effect of temperature [10]–[12]. In this paper, an extensive review of existing ultrasonic measurement techniques for liquid level monitoring is presented and a novel Artificial Neural Network (ANN) technique to compensate the environmental perturbations that affect the accuracy of ultrasonic measurement is also proposed. The main objective of this work is to improve the accuracy of ultrasonic based measurement system, which is adaptive to the environmental changes in the measurement medium. We propose a modified ANN based UMS, which can minimize the error substantially. The same model is also used to extend the operating range of the ultrasonic sensor.

3.1 The components/hardware used are:

1. **Arduino UNO** – The micro controller that processes the data obtained from the sensors and send necessary information when connected to a communicative device/module
2. **Ultrasonic sensor (HC-SR04)** - An ultrasonic sensor 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.
3. **Temperature and Humidity sensor (DHT22)** – It is used for sensing temperature and humidity. It gives an analog voltage output that can be processed further using a microcontroller
4. **Bluetooth module (HC-05)** – A connectivity module that can be used to communicate in between a micro controller and a smartphone/PC
5. **Mobile/PC** – These are the devices used to receive information from the Microcontroller

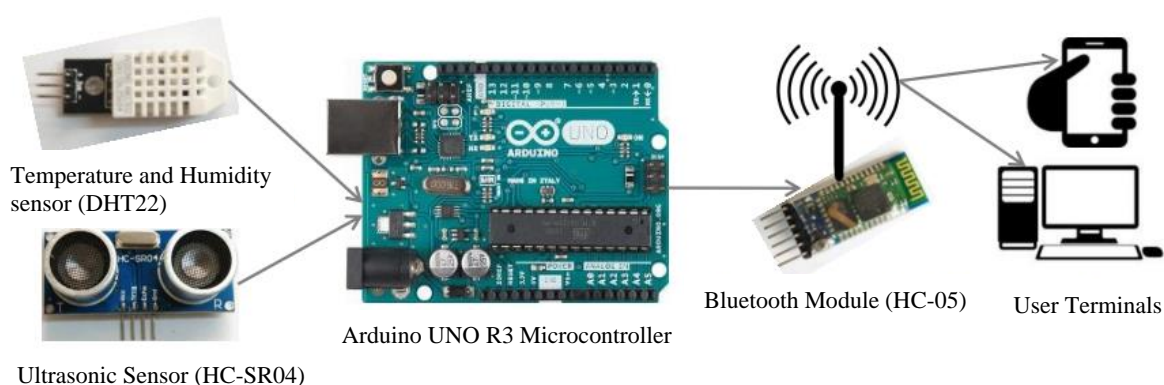


Fig. 3.1: Basic block construction of the existing system.

3.2 SYSTEM IMPLEMENTATION:

The function of this system shows the estimated volume of the liquid, when the device is calibrated according to the environment of the container/entity and placed on the top of large water containers like tanks, wells, etc.

1. This device powers on a standard voltage of 5V and can be powered in multiple ways
2. All the components (sensors) used in the project also used the same standard voltage of 5V. The power drawing capacity varies on the application and usage.
3. This system also considers the effect of temperature and relative humidity on sound waves. As the speed of the sound is relatively equal to the speed of the ultrasonic waves, they considered this.
4. Uncertainty in ToF ($u^2(T_f)$) is mainly because of the influence of noise and attenuation. Uncertainty in the measurement is introduced due to
 - (i) The temperature gradient in the measurement path,
 - (ii) Humidity gradient (this is more in the case of water tanks),
 - (iii) presence of other gases,
 - (iv) electronic noise (due to electronic circuit imprecision). Water storage tanks located outdoors are directly exposed to sun light and experience a temperature swing and hence, there is a variation of temperature and humidity inside the tank.

A sample observation of change in temperature and humidity level from the point of measurement to the water surface is shown in Table below. It is not possible to accurately compensate the speed of sound wave in air medium considering all the above factors simultaneously. Thus, in this work, we propose an ANN based approach to address the uncertainty issues in low-cost ultrasonic sensor-based level measurement.

Table 3.1: Variation in temperature and relative humidity at three different positions for different storage tanks.

Depth (d)	Top position		Middle Position		Near water surface	
	Temp (°C)	Humid (%)	Temp (°C)	Humid (%)	Temp (°C)	Humid (%)
100 cm	24	71	23.5	72	23	73
150 cm	30	48	30	51	29	54
200 cm	33	61	33	63	32	64
250 cm	37	54	36.3	57	35.2	59
300 cm	32	68	31	68	30	73
350 cm	34	56	33	58	33	62
400 cm	41	36	40	38	38	42
450 cm	36	44	35.5	47	33	51

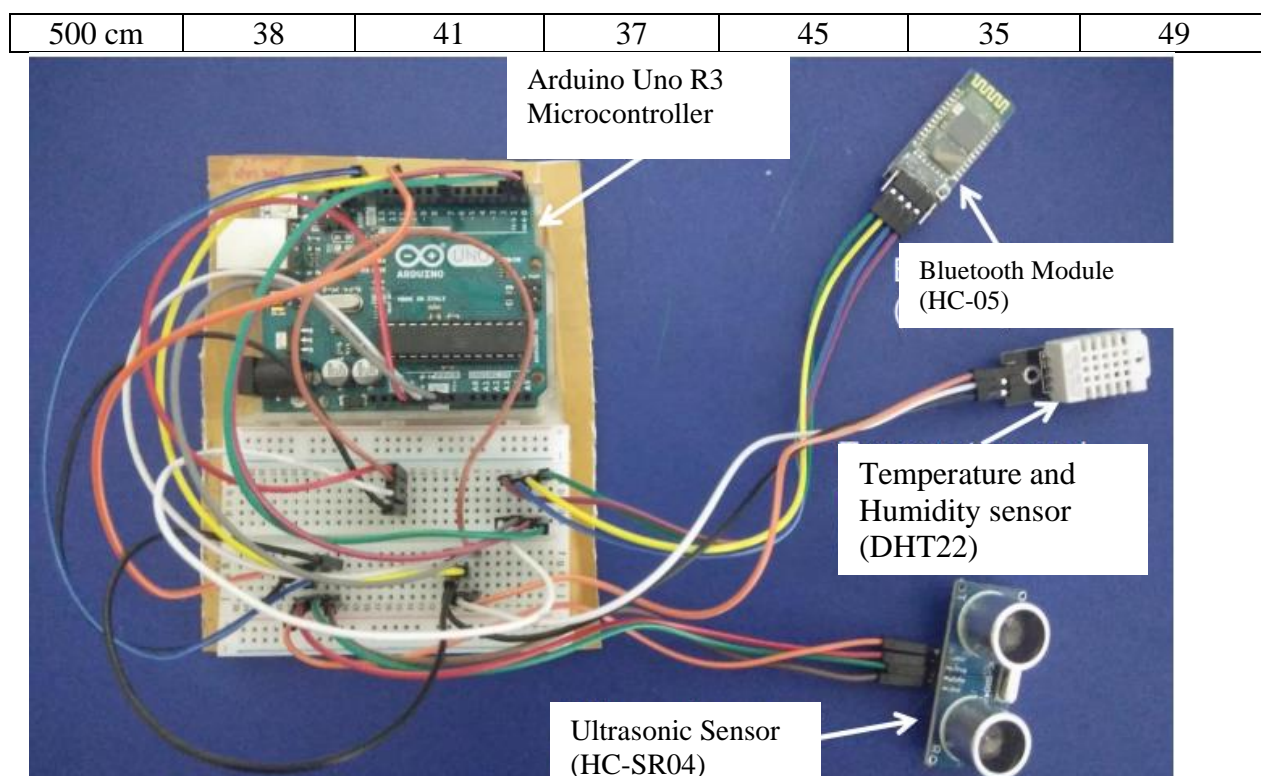


Fig 3.2: The physical connections of ultrasonic sensor, temperature and humidity sensor and Bluetooth module with the Arduino UNO micro-controller.

3.3 Testing and evaluation of the model with new data sets

The performance of the system's neural network model is further evaluated with a new data set. The new data set which consists of 508 data points is divided into five segments to evaluate the performance of all five trained neural sub-networks. For segment-I, there are 111 data points for input to the neural sub-network-I and the output estimated distance error variation lies within ± 0.19 cm which is around 0.2 %. Similarly, segment-II contains 91 data points, segment III contains 92 data points, segment-IV contains 110 data points and segment-V contains 104 data points. The estimated distance variation for all of these segments is around 0.3 %. Figs. 13(a)-(e) show variations in output of the proposed model and actual distance measurements of all five segments. The results demonstrate that proposed model works better when there is a variation of temperature and humidity between the UMS and target surface which is observed in real experimental scenario. A sample of experimental observations to depict the variation in temperature and relative humidity in the measurement medium is shown in Table VII. Table VII shows the variation of temperature and relative humidity measured at top, middle and near to water surface of storage tanks of different depths (100 cm to 500 cm). From this table, it is observed that there is a variation of temperature and humidity level between the point of UMS installation and near to water

surface. Although the ultrasonic sensor data sheet mentions that the operating range of distance measurement is restricted to 400 cm, we have tried and used this model for extended distance range up to 500 cm. Beyond 400 cm, the error observed was large in the standard theoretical model, but the ANN model is able to compensate and reduce the distance measurement error to an acceptable limit of ± 1 cm. Thus, the second objective of extending the operating range with the use of proposed model without upgrading the hardware is also achieved.

3.4 Implementation:

The experiment was conducted at temperature ranging from 22 °C to 45 °C and relative humidity ranging from 25 %RH to 80 %RH. The level measurement readings were taken at different water levels ranging from 2 cm to 500 cm in different storage tanks. Experiments were also carried out at different time intervals during day time to acquire variations of temperature and humidity data in the measurement medium. The data from the ultrasonic sensor, temperature sensor, humidity sensor, computed speed of sound and measured distance for different water levels were taken and logged to the machine through communication module (HC-05) for analysis. The speed of sound was computed using (6). Measured distance or ultrasonic distance was computed using ToF and speed of sound. Actual distance measurements were taken using a metric ruler with a resolution of 1 mm. The UMS was installed on water tank at top center position with the ultrasonic transducers facing downwards, perpendicular to the water surface as shown in Fig. 9. Distance measured by the developed UMS is denoted as estimated distance. To evaluate accuracy of the UMS, experiments were conducted in different environmental conditions. Experiments were also conducted by blowing hot air into the medium to check the effect of change in temperature, humidity, and ToF of ultrasonic waves in level measurement.

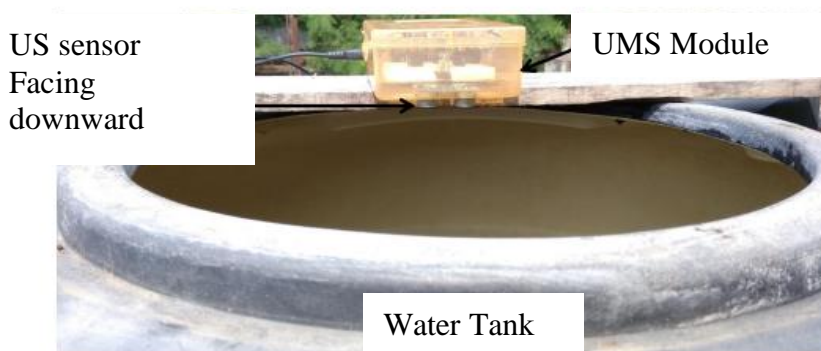


Fig. 3.3: Snapshot of experiment conducted on a water tank to measure water level using the UMS module

3.5 Discussions:

It may be noted that the developed ANN model was trained offline and thereafter, the model was implemented in the UMS for online level or distance measurements. This developed model reduces the error from 1 % to 0.3 % and is also able to extend the distance measurement range by another 100 cm, which is almost 25 % increase in the operating range of the used ultrasonic sensor. The method can be considered as a substantial improvement in many industrial applications such as measurement of high-cost liquid level, where the required distance measurement accuracy is in millimeter ranges. In general, when the storage tank diameter is more, a small variation in level measurement will lead to incorrect information about the volume of water contained in the tank.

The response time of temperature and humidity sensor is 2 seconds as per the data sheet of DHT22 (AM2302). But in real time we found that the response time of DHT22 is around 380 milliseconds. The execution time of speed of sound formula given in (6), ToF of ultrasonic sensor and distance measurement computation is around 5 milliseconds. The response time of the ANN model is around 2 to 3 milliseconds. The overall response time of the developed UMS model is around 400 milliseconds. The ultrasonic sensors are the best option for non-contact distance or level measurement as they are accurate, safe and low in cost when compared to other noncontact sensors. Some cases such as agitated liquids, turbulent liquids, foaming, sloshing and similar other phenomena, and liquids that absorbs ultrasonic waves instead of reflecting, can adversely affect the performance of ultrasonic sensors. In these cases, the ultrasonic sensor based level measurement may not be applicable. In some cases like agitated and turbulent liquids, an average of multiple observations may be considered. The proposed ANN based low-cost UMS module can be used to predict and detect flash floods as it is able to continuously monitor the level. It can also monitor other industrial applications where higher accuracy is a major requirement. The same model can be used for different applications and environments by retraining the ANN model and the weights can be relearned before putting it to use. As the training is always off-line, it will not affect the actual measurement time and will not add to the computational complexity of UMS.

3.6 Conclusions and Future scope:

This study proposed a modified ANN architecture model for accurate prediction of distance using UMS with a special case study for water level measurement in water tanks. The experimental findings demonstrate the effectiveness of the model in making the UMS adaptive to find accurate distance in different environmental conditions. The ANN produces

the best-minimized error when an appropriate combination of model parameters are used for the training process. In addition, we have proposed an ANN model consisting of five neural sub-networks for different range of distances to reduce the error across the distance measurement ranging from 2 cm to 500 cm. This model not only measures the water level but can also be used for other applications where manual and contact based distance measurement is difficult or not possible. The developed adaptive UMS model compensates all major environmental parameters that affect the ultrasonic sensor measurement to get better accuracy. The proposed model also extends the operating range of the used ultrasonic sensor from 400 cm to 500 cm with controlled error in measurement. In future, we want to extend our work for level measurement of liquids like petroleum products, turbulent water, and surfaces that absorb the ultrasonic waves partially. The finding of our current work will be further used to test other algorithms and models to improve the performance of the UMS to accurately measure long-range distances in different industrial applications.

CHAPTER 4

PROPOSED SYSTEM

The proposed system consists of a hardware platform that integrates various electronics and an algorithm suite that uses sensor data for fluid intake monitoring. Our state machine and algorithms for bottle state recognition, volume estimation, and fluid type detection are discussed thoroughly.

4.0 Design Considerations:

The major design principle of this system is supporting the usability of the system for everyday use. People use different bottles or cups for drinking water or other fluids in a daily basis. Our principle is to design a system in such a way that it can be used for different containers. For instance, most people may drink tea in a cup and drink water from a bottle. As a result, having a system which can be used for both scenarios is desirable. Furthermore, we envision a sensing system that is low-cost, small in size, and light-weight, such that it can be conveniently installed as a lid on a fluid container. Having the ability for LIDS to be used with different bottles in different conditions brings the challenges associated with the adaptability and robustness. As the ultimate goal of our system is to be deployed in end-user settings, we need to assure that it is effective in such settings where uncertainties due changes in the reusable bottle (e.g., bottle color) and environment (e.g., ambient light) are taken into consideration during the system design.

Moreover, computational models need to be able to utilize the movement readings to effectively distinguish the fluid intake action from other movements such as refilling the bottle that may occur. In addition, the system needs to correctly classify fluid intake activities from many different users who may have varying signal patterns and signatures. LIDS needs to adapt itself to different conditions such as different brightness conditions, or using LIDS with transparent, translucent, or opaque containers.

As a result, based on RGB color sensor, we propose a comprehensive data collection process which includes different scenarios such as environments with different light intensities, different fluid volumes and different bottle colors. Power efficiency is another concern which is vital for a health monitoring system that is going to be used throughout the day. Therefore, the minimization of the energy consumption while maintaining the performance is crucial in designing LIDS. Energy efficiency is important to ensure long-term continuous functionality of the system and enhance adoption of the system.

In particular, the system collects and processes various sensor data including an ultrasound and RGB sensors which consume a significant amount of power. As a result, we propose a signal processing methodology based on the accelerometer sensor that identifies the optimal times for volume and fluid type detection to drastically decrease the duty cycle of the sensors by putting them in sleep mode while not in-use.

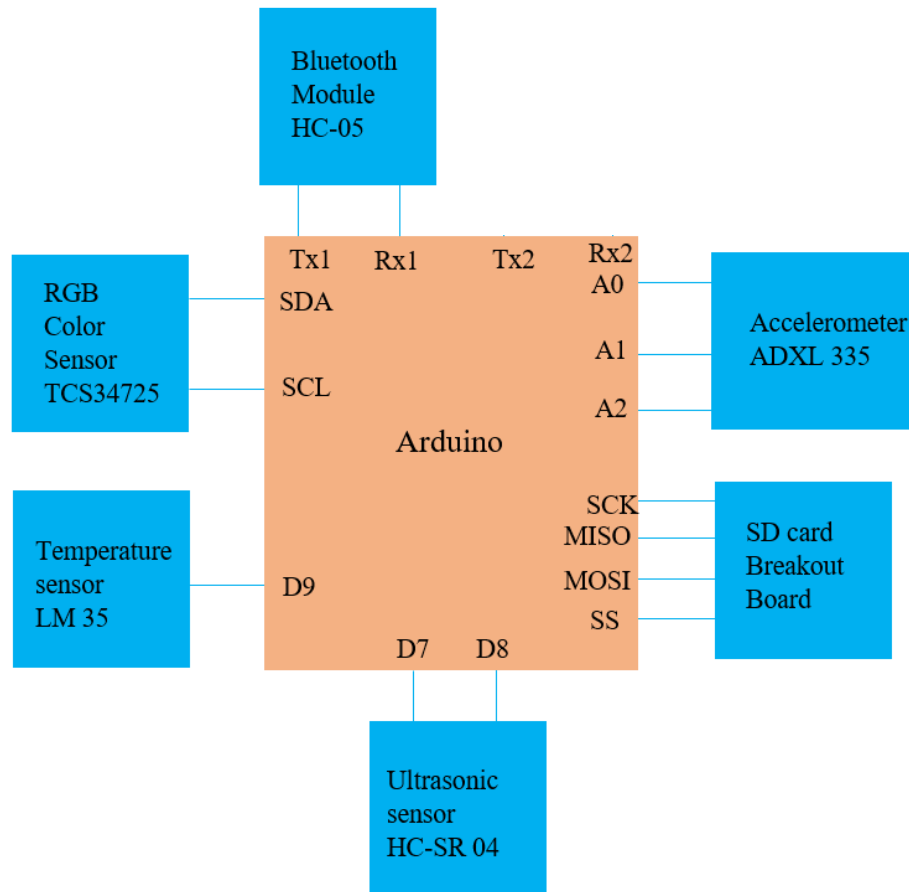


Fig 4.1: Electronic components of proposed system.

This includes an array of sensors

The array of sensors:

- Ultrasonic sensor
- RGB color sensor
- Infrared temperature sensor
- An accelerometer to monitor the volume and type of the fluid inside the container, as well as important motions related to drinking activities (i.e., stationary, closing the lid, drinking from the bottle, or ambulating)

4.1 Hardware Platform:

Fig 1 shows various hardware components used in designing the proposed system. The diagram also shows the circuitry of the hardware components and how each module is connected to the main controller. We utilize four different sensors including accelerometer (ADXL335), ultrasonic (HC-SR04), RGB color (Adafruit TCS3200), and temperature (DS18B20) sensors. Most of the electronic components are installed inside a box to protect them from water and other fluid.

However, sensors such as RGB color, ultrasonic, and temperature sensors remain outside of the box to monitor the fluid. We use a waterproof ultrasonic sensor, shown in Fig 1, to ensure sensor operation in proximity of the fluid. We use epoxy to cover the circuits of RGB color sensor and temperature sensor to prevent the direct contact between the fluid and the sensors. Following our design principles, all the sensors need to function on a lid of a bottle and in a non-contact way. Therefore, for volume measurement through lid of a bottle we need a non-contact proximity sensor. There are three types of noncontact proximity sensors including capacitive, photoelectric, and ultrasonic sensors. However, capacitive sensors' sensing range falls short for our application. Their typical measurement ranges are between 1 and 60 millimeters. In addition, the shape and the color of the object can affect the readings of photoelectric sensors. Therefore, we choose ultrasonic method which can measure distance up to 3 meters. For temperature measurement, we use DS18B20 which is an infrared thermometer designed for non-contact temperature sensing. An Arduino micro-controller samples each sensor and transmits the sensor readings via Bluetooth (Sparkfun Bluetooth Mate Gold) to a mobile device for data processing. LIDS is also capable of logging the collected data on an SD card (via Adafruit Micro-SD card breakout board). A rechargeable LiPo battery with the capacity of 500 mAh is used to power the entire hardware board. In order to construct an affordable prototype for fluid intake monitoring, all the components of the sensing system cost less than two thousand.

4.2 Method of operation:

In this section we explain the design flow of LIDS and how it monitors fluid intake. We begin with an overview of the various procedures and computational tasks, which are needed for LIDS to compute fluid intake features. We then focus on each computational task and describe bottle-state recognition, volume estimation, and type detection algorithms.

4.2.1 Design Flow:

Fig 2 shows the overall state machine of LIDS that indicates different states of LIDS during the regular operation. However, there are two main processes including calibration and regular operation.

A. Calibration:

Before using the system for fluid intake monitoring, there exist a calibration process during which (1) the user indicates the diameter (R) of the bottle through a mobile application user interface; (2) the system performs a self-calibration to measure the height (H) of the bottle while the bottle is empty; (3) the system records the values of RGB color sensor of the empty bottle for further calculation which helps us to estimate the color of the bottle. This calibration process is accomplished through a mobile device (e.g., smartphone) and the sensors and algorithms embedded in the bottle. First, the bottle-state recognition algorithm identifies if the bottle is in an stable position during the calibration process. The ultrasonic and RGB color sensors are then sampled to measure the height of the bottle. Therefore, the process of volume estimation assumes that the height of the bottle is known a priori. Because of the portability of the sensing components, sensors can be mounted on the lid of cylinder-shaped bottles with various heights. For the first use, the bottle needs to be empty and stable to measure the height (H) of the bottle.

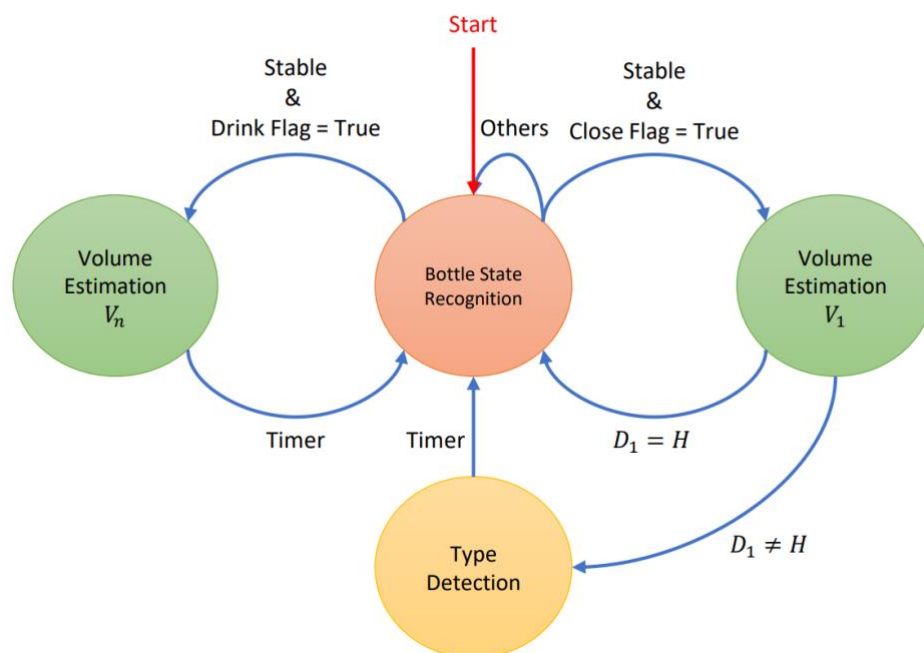


Fig 4.2: LIDS operational state machine during both calibration phase and regular operation.

These states include bottle-state recognition, volume estimation, and type detection. Bottlestate recognition controls the system by setting the flags (Drink and Close Flags). These flags activate other functions for volume estimation and type detection.

B. Regular Operation:

The regular operation of LIDS is designed to sample each sensor based on the events of interest that are detected by LIDS movements. Therefore, LIDS samples when there is a potential change in fluid volume. Furthermore, it ensures that LIDS doesn't miss an event that can lead to a change in fluid volume and type. LIDS collects accelerometer data for classifying LIDS movements and utilizes ultrasonic, RGB and temperature sensor data for fluid intake monitoring. The system stores the volume of the fluid intake and their type continuously. After a pouring event which is detected by closing LIDS event (Algorithm 1), LIDS waits for a stable event to read other sensors values. LIDS first estimates the volume of the poured fluid and then it detects the fluid type. However, if the estimated volume is equal to the bottle volume LIDS goes back to its initial state because the bottle is empty. We conclude that closing event probably happened for other reasons (e.g., washing) rather than pouring a fluid into the bottle. If LIDS detects that the user actually poured a fluid into the bottle by the volume difference from empty bottle then it activates the fluid type detection function. After capturing initial fluid type and volume, LIDS waits on detecting a drinking event (drink flag on Fig 2 and Algorithm 1). This event indicates how much of the poured fluid the user consumed from the bottle. However, if before drinking event a closing event happens then LIDS assumes that the user pours a new fluid without drinking the last poured fluid. After drinking event is detected, LIDS waits for the LIDS to be in a stable position to estimate the consumed volume. Ultimately, LIDS reports the consumed volume and type of the fluid via mobile application. LIDS do not assume that the user drinks the whole poured fluid. Therefore, it can also report that a certain amount of volume which is not the initial poured volume is consumed by the users. In the following sub-sections, we elaborate on the machine learning algorithms devised in LIDS.

Algorithm 1 Fluid intake monitoring algorithm

```

BottleState ← BottleStateRecognition(Accelerometer)
while BottleState = Stable do
    if CloseFlag = True then
        D1 ← UpdateD(Ultrasonic, RGB)
        if D1 != H then

```

```

        Type  $\leftarrow$  TypeDetection(RGB)
        CloseFlag = False
        else Report  $\leftarrow$  UpdateReport(Volume, Type)
        end if
    else
        if DrinkFlag = True then
            D2  $\leftarrow$  UpdateD(Ultrasonic, RGB)
            d = D2 - D1
            Volume =  $\pi R^2 d$ 
            DrinkFlag = False
            Report  $\leftarrow$  UpdateReport(Volume, Type)
        end if
    end if
end while

```

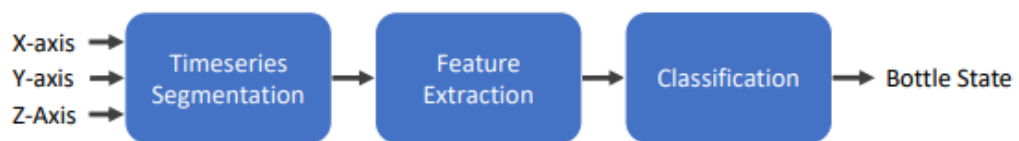


Fig 4.3 : Signal processing and machine learning flow of the supervised algorithms in LIDS for bottle-state recognition.

The inputs of the process are the accelerometer data and the output is the bottle state (close, drink, stable, and others).

4.2.2 Bottle-State Recognition:

For detecting bottle states, we focus to four classes/states including “stable”, “close”, “drink”, and “others” activities. These class labels refer to four states of the bottle including when the bottle is stable, when the user is closing the lid, when the person is drinking from the bottle, and when the bottle is not stationary, respectively. Detecting these states are required for performing fluid intake monitoring. The high-level data processing flow for bottle-state recognition from accelerometer signals is shown in Fig 3. This figure shows signal processing and machine learning algorithms used in bottle-state recognition approach. The inputs are the accelerometer readings for x-axis, y-axis, and z-axis. We use a low-pass filter to filter high frequency noise from the accelerometer time-series, segment the time-series using a sliding window approach, extract statistical features, and train a classification model

for bottle-state recognition. For segmenting the time-series data, we apply a sliding window of 10 seconds with 2 seconds overlap between the adjacent windows. Extracted features include the minimum, maximum, mean, standard deviation, root mean square power and peak-to-peak amplitude of the incoming signal within each sliding window (i.e., signal segment). We use Recursive Feature Elimination (RFE) method which is a wrapper-type feature selection algorithm to rank features for selection. Random Tree classifier is used as the machine learning algorithm for ranking features. Ground truth labels (i.e., correct bottle states) are gathered during the data collection process. The labeled feature vectors are fed into a machine learning engine for algorithm training. We use a Fit Binary Classification Decision Tree for multiclass classification to classify the bottle state.

4.2.3 Volume Estimation:

When the system detects a “closed” event followed by placing bottle in an upright position (i.e., “stable” event), it calculates the initial distance between the cap and fluid surface (D1). As shown in Figure 5a, LIDS measures the distance between the sensor and the fluid surface (D). Based on the values of D and H, the system measures the volume of the fluid left in the bottle. When the system detects a “drink” event, again it estimates the new distance. This new distance value is referred to as D2, which is the new distance between the lid and fluid surface. The consumed amount of fluid, d, is calculated as the difference between D1 and D2 (Algorithm 1).

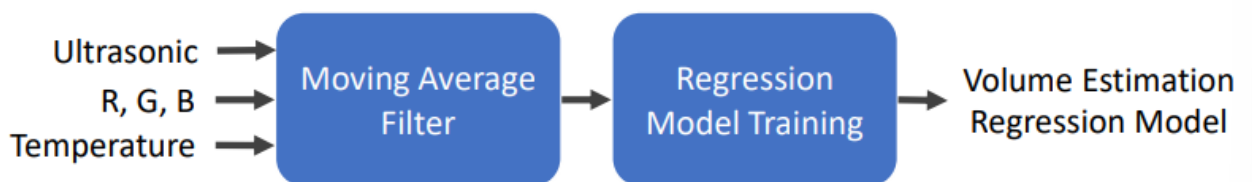


Fig. 4.4: Machine learning flow for volume estimation.

The inputs are the sensors readings and ground truth volume levels. Moving average filter is used on the collected data as a low-pass filter. A regression model is trained to estimate volume levels.

Previous research suggests that different fluid types reflect different ultrasound signals that depends on the fluid properties, such as density and viscosity. As a result, raw ultrasound signals may not be able to measure the distance precisely for different fluid types. Therefore, according to Fig 4, we gather ultrasonic, RGB and temperature data with ground truth labels during a comprehensive set of experiments using different volumes of different

fluids. Moreover, the sensors data undergo a moving average filter which is a technique that can be used to remove noise from the signals. Then we feed the training set to a machine learning algorithm that generates a model for volume estimation. We utilize widely used learning algorithms including Linear Regression, Polynomial Regression, Random Forest, Support Vector Machines, k-Nearest Neighbors, Naïve Bayes, and Random Tree. We choose the one that generates the most accurate prediction. These algorithms can be used for both regression and classification problems. For all algorithm we set the batch size to 100 samples. For Random Forest the number of trees is set to 100. We set maximum depth of the tree to unlimited. For Linear Regression algorithm we utilized greedy method as attribute selection method. We also set algorithm to eliminate colinear attributes. The ridge parameter for this experiment is set to $1.0e - 8$. For k-Nearest Neighbors we set the number of neighbors (k) to 1. For Random Tree we set minimum number of instances per leaf to 1 and we also set minimum numeric class variance proportion of train variance for split to $1e - 3$. Maximum depth of the tree is set to unlimited.

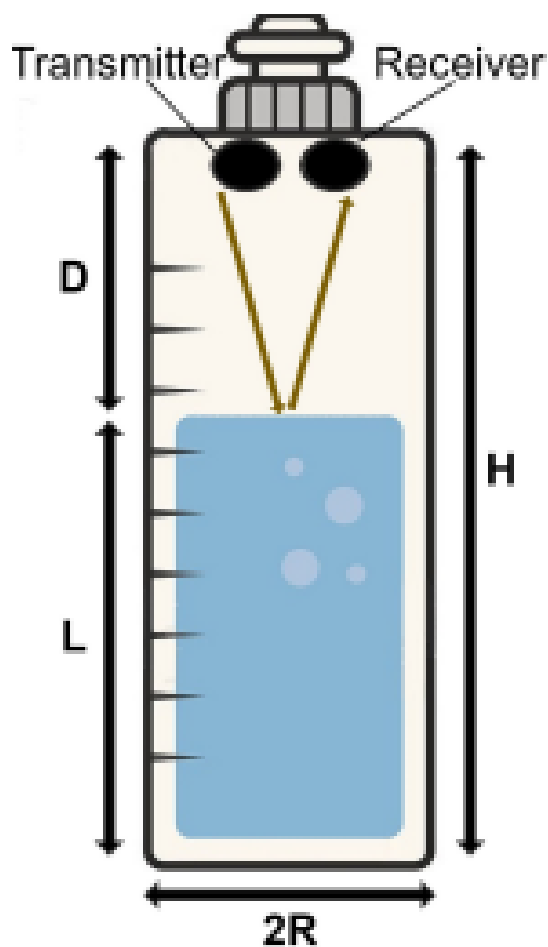


Fig 4.5: Volume measurement procedures.

The figure shows the main approach for volume estimation which is based on transmission and receiving ultrasound signals.

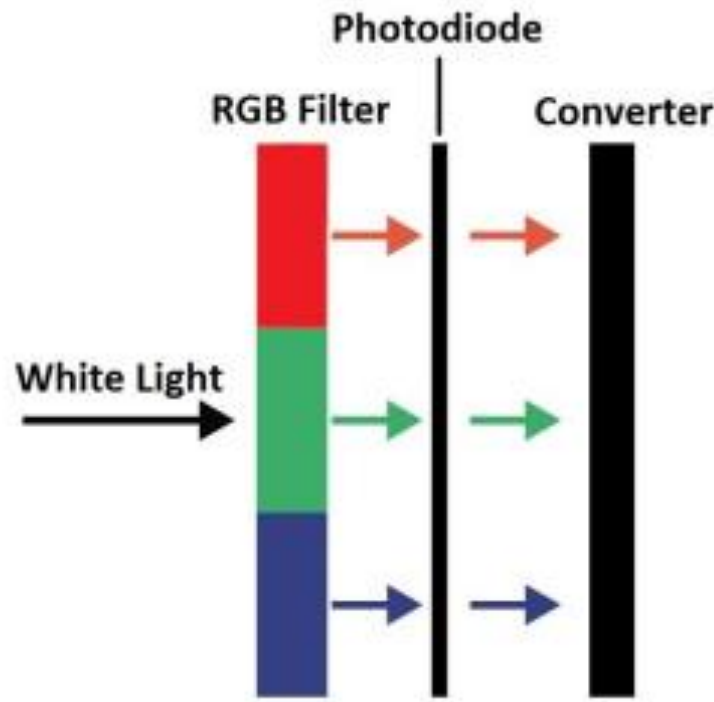


Fig 4.6: Color detection with RGB color sensor

The above figure shows the main process of fluid type detection which is based on shining a white light at fluid surface and recording the reflected color through RGB filter.

4.2.4 Temperature Compensation and the speed of sound

If you have been using Arduino, you might have used or have heard about the ultrasonic sensor. Most of us use it for level/distance measurement while some use it to detect an obstacle in front. What most of us don't know is that there is an underlying flaw with the ultrasonic sensor, particularly on the first application I mentioned (level/distance measurement).

The ultrasonic sensor is able to detect an object in front of it by emitting an ultrasound and then waiting for that ultrasound to bounce back. Naturally, the further the object, the longer the "echo" to arrive.

Most tutorials , uses this formula to determine the distance of the obstacle:

$$\text{distance} = \text{duration} * 0.034 / 2;$$

Here, 0.034 is the speed of sound in centimeters per microsecond. So when the “echo” returns in 100 microseconds (t in the formula), the distance of the obstacle that bounced the ultrasound is 1.7 centimeters.

The Calculation:

The error lies in the value of the speed of sound because this value is actually temperature dependent. In fact, the practical formula of the speed of sound is:

$$C_{\text{sound}} = 331.3 + 0.606T$$

Where T is the temperature of the surrounding in degree Celsius. Moreover, the usual speed of 0.034 cm/us or 340 m/s, using this formula, only works when the temperature is 14.35 °C! Now let's see the measurement difference between the two formulas. The old one, as mentioned, detects the obstacle to be at 17 centimeters. For the new formula, I will be using today's temperature which is 33 °C (yeah it's hot here). This gives me a speed of sound value of

$$331.3 + 0.606(33) = 331.3 + 19.998 = 351.298 \text{ m/s}$$

Consequently, the distance of the object would be:

$$d = \frac{351.298 \times 10^{-3}}{2} = 17.56 \text{ cm}$$

A 0.56 centimeter difference! Also, this difference may vary during different times of the day as the temperature varies.

4.2.5 Type Detection:

RGB color sensor data is the main source for fluid color detection, which then helps to recognize the fluid. Fig 5b shows the flow of the sensor RGB color sensor. Utilizing red, green, and blue color filters, the photodiode converts the amount of light to current. The converter then converts the current to voltage which the controller unit reads. Fluid type detection can be challenging due to different colors of bottles which affects the RGB sensor readings. This process requires the color bottle information which is gathered during the calibration process. There are different bottles with different colors such as yellow, blue, red, transparent, black, etc. Therefore, using bottles with different colors can challenge the capability of our type detection method. We assume that users may use an unseen bottle to the system. To address this challenge, we record the r,g,b values of the new bottle when it is empty during the calibration phase. Then, we define three variables including rc , gc , bc .

Equation 1 shows the equations that we use to calculate the mentioned variables in calibration phase. These values aim to reduce the potential effects of light intensity of the environment. The summation of these values is 1 and for instance if r_c is the greatest between these values, this indicates that the container is more reddish. Therefore, we expect to observe the effects of red color on each fluid more than the other colors. In fact, these variables help us to estimate the color of bottles which can provide some information about the potential effects of the bottle color in each fluid. For each in-use bottle we compute these variables. Then we utilize two different methods for fluid type detection. These methods include

- 1) direct supervised learning approach similar to our process for bottle-state recognition using classification and regression models and
- 2) feature embedding approach which uses neural network for type detection.

$$r_c = \frac{\sum_{i=1}^N R_i}{\sum_{i=1}^N (R_i + G_i + B_i)} \quad \dots\dots\dots (4.1)$$

a. Direct Supervised Learning Method:

In this method we construct our dataset with raw sensor readings (red, green, blue, proximity, light intensity, fluid temperature) and bottle calibration values (r_c , g_c , b_c). The class labels are the fluid types that we aim to predict. We gather sensor data including ultrasonic, RGB color, light intensity and temperature sensors with ground truth labels during an extensive set of experiments using various types of fluids and bottles with different colors. We then feed the training dataset into different machine learning algorithms that generate models for detecting fluid type. These machine learning algorithms include Support Vector Machine (SVM), k-Nearest Neighbors (kNN), Random Tree and Random Forest. For all algorithm we set the batch size to 100 samples. For SVM, we set the complexity constant "C" to 1. We also set the tolerance parameter to $1.0e - 3$. The algorithm is set to utilize polynomial kernel with the cache size of 250007. Logistic regression is set to act as SVM calibrator in this experiment. The tuning parameters that are set for k-Nearest Neighbors and Random Forest is the same as the volume estimation. LIDS uses a Random Forest classifier to detect the fluid type using sensor values and uses the output for generating the reports shown in Algorithm 1.

b. Feature Embedding:

Unless the train and test data are drawn from the same distribution, supervised learning methods perform poorly at making predictions. This problem is known as out-of-distribution generalization. Because of this reason, the direct classification method may not be generalizable for unseen bottles. One potential solution would be to collect a large labeled dataset of fluids in the unseen bottle and re-train the model, which might not be feasible in the real-world use cases. To tackle this problem, we utilize the feature embedding technique to find a shared color space between all bottles. In this approach, we aim to find an embedding space such that the classification accuracy for unseen data is maximized. Fig 6 shows the proposed solution for color embedding. First, we transform the colors of input space into the color space, and the output of the color space (output of embedding) is used for fluid classification. Intuitively, we try to transform the color-dependent features (e.g., r, b, rc, bc, ...) into a shared space and thus find a representation for color features. This is shown in the red box of Fig 6. The transformed color features, in addition to the other features including temperature (t), proximity (p), and light intensity (l) will be the inputs of the classification problem, as shown in the yellow box of Fig 6.

To implement this approach, we use a differentiable mechanism. Fig 6 represents a feed-forward neural network with neurons and connections shown by circles and lines, respectively. The objective function is the classification accuracy measured by cross-entropy loss. For the neural network architecture, we use three hidden layers, each with 50 ReLU neurons, and used Adam with a learning rate of 0.005. Moreover, we have used the Dropout technique for regularization with the dropout probability of 0.25. The used hyper-parameters were calculated using a grid search on number of neurons in each layer (25, 50, 100), learning rate (0.0001, 0.0005, 0.001, 0.005, 0.01), and dropout probability (0.1, 0.25, 0.5). Our proposed method has two advantages: First, by integrating the problem of finding the shared embedding space and the classification problem, we need to solve only one optimization problem, and this reduces the complexity of our system. Second, by using the neural network, we remove the need for feature engineering.

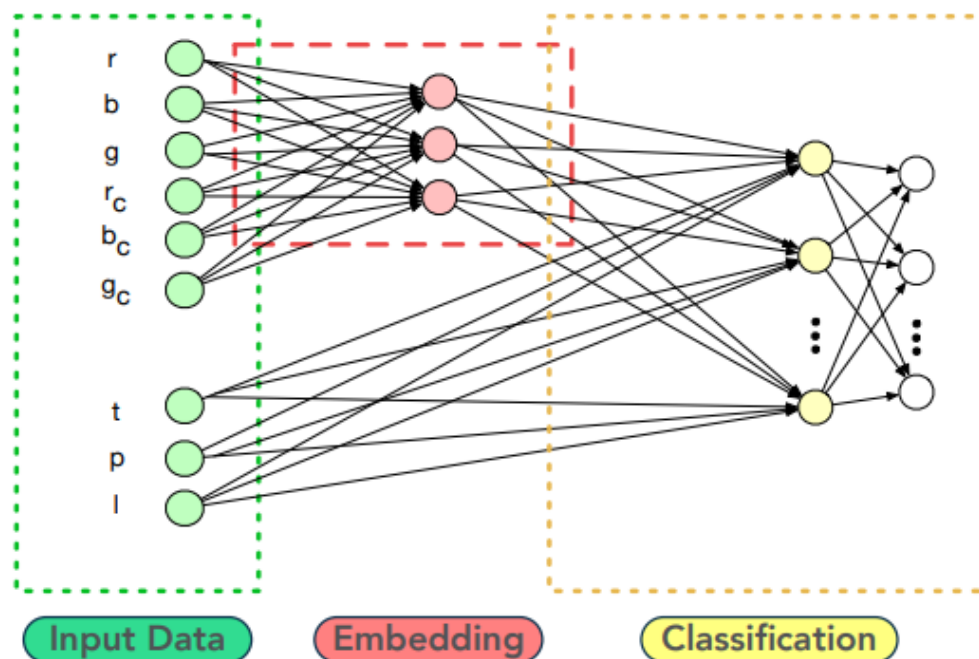


Fig 4.7: Proposed neural network for color embedding. Inputs are the sensors readings and bottle color. Outputs are the fluid types which achieve through classification.

4.2.6 Performance Metrics:

For fluid volume estimation we compute relative absolute error (RAE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Root Relative Squared error (RRSE) of different regression models. We compute MAE and RMSE as metric to report the volume estimation performance. MAE is used to evaluate the overall performance of the fluid volume estimation by describing the average error. However, we also use RMSE because it gives a relatively high weight to large errors. Since large errors are $\dots\dots\dots$ (4.2) fluid volume estimation. On other hand, we also compute the relative error of both metrics to report the percentage of the error over the capacity of the tested bottles which helps us to determine the performance of fluid volume estimation for bottles with different sizes. The equations in (2), (3), (4) and (5) give these performance metrics where a , a_0 and a_m denote actual values, estimated values, and the mean of actual values, respectively. We use accuracy (6) as our performance metric for bottle-state recognition and fluid type detection which are classification problems with k number of classes. Accuracy is defined as the number of the correctly classified instances which includes True Positives (TP) and True Negatives (TN) over the total number of instances of the dataset including True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN).

$$MAE = \frac{1}{N} \sum_{i=1}^N |a'_i - a_i|$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (a_i - a'_i)^2} \quad \dots\dots\dots (4.3)$$

$$RAE = \frac{\sum_{i=1}^N |a'_i - a_i|}{\sum_{i=1}^N |a_m - a_i|} \quad \dots\dots\dots (4.4)$$

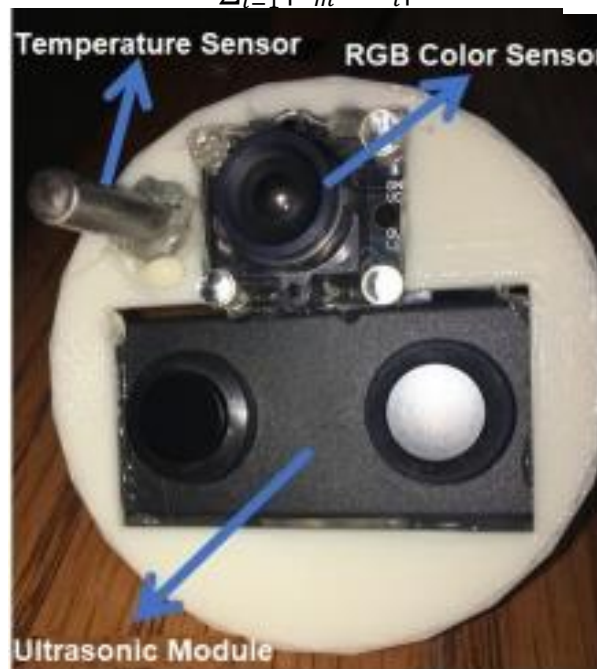


Fig 4.8: Current LIDS prototype

4.3 Data Collection

A smartphone application was developed for data collection. The application was used to gather sensor data in real-time and to transmit the data to a computer where our machine learning algorithms are trained. The user interface allows to visualize the fluid type and volume as the user may pour fluid into a bottle. The mobile application serves as a base station and provides an interactive user interface as well.

C programming language is used in Arduino Integrated Development Environment (IDE) to read sensor data and communicate with the base station. We collected data on four different bottles including blue, red, opaque, and transparent bottles (Fig 7b). The data collection is done in a lab environment. We collected data for five different volume levels (0 mL, 150 mL, 300 mL, 450 mL, and 600 mL) and ten fluid types (water, orange juice, lemon juice, apple juice, cranberry juice, black tea, coffee, coke, milk and chocolate milk). In order

to account for differences in light intensities, we collected data in three different light settings including dark (0 lx), medium light (200 lx), and extreme light (1000 lx). In summary, the constructed dataset from our experiments on each bottle included three light settings, five volume levels, and ten fluid types, which resulted in a total of 150 different data collection trials and a total of 600 data collection trials for four used bottles. We repeated each trial two times to enhance the reliability of the data collection, resulting in 1200 data trials. In addition, we collected data for bottle-state on three subjects. We had one female and two male subjects, and the age range of the subjects were between 24 to 30. The data collection includes mobile LIDS, leaving LIDS on a surface, drinking, and opening/closing LIDS. A human subject observing the experiment labeled the sensor data annotation to indicate closing, drinking, stable and other events. Subjects also carried the bottle in their bags and cars for mobile scenario, which were labeled as others class. Each subject was asked to use the bottle for drinking, opening/closing. They also asked to leave the bottle stationary and move with the bottle. Each trial repeated 20 times for each subject which resulted in 240 data collection trials for bottle-state recognition. Each trial lasted for approximately 60 seconds and all the sensors were sampled at 10 Hz. We carefully designed data collection scenarios to assess generalizability of our algorithms. To this end, we included various experimental conditions that result in highly distinct sensor readings for each condition.

4.4 Volume Estimation

The low-pass filter (moving average) is programmed in the microcontroller using C programming language while sampling sensor data in real-time. We utilized Weka data mining toolkit to train the regression model and investigate volume estimation performance. First, we use the collected data to build our regression models. Second, we evaluate our model based on unseen volume levels and fluid type.

4.4.1 Regression Models Comparison:

We compared the obtained results from Random Forest Regression with Linear and polynomial Regressions which are three popular regression models. To evaluate the model on the collected data, we used 10-fold cross-validation. Table I shows the RAE, RMSE, MAE, and RRSE of the Random Forest regression model compared with the Linear and Polynomial Regression models. Results in Table I indicate that Random Forest Regressor outperforms both Linear Regression and second order Polynomial Regression models for this application. Fig 8 and Fig 9 compares the ground truth with the model output. Blue line in Fig 8 shows

the Ground Truth (GT) while the red line is predicted by the Random Forest (RF) model. It can be seen from Fig 9 that LIDS works more accurately while the volume of the bottle is lower, which can be justified by the fact that ultrasonic module range is limited. The minimum practical distance measurement of ultrasonic module is 2cm which makes the readings inaccurate while the distance is around this value.

4.4.2 Leave One Volume Out (LOVO):

We recognize that we modeled the problem of volume estimation as a regression problem where the dataset included only a small number of volume levels. This may raise a question of whether or not our model is generalizable to previously unseen volume levels. To investigate this issue, we conducted an experiment, referred to as the leave one volume out experiment, where we train a model using all volume levels except one test level. We train the regression model with four levels and tested the model with the volume level that has been removed from the training data. As shown in Table II, the amount of Relative Absolute Error (RAE) ranges from 1.52% for highest volume out (i.e., volume level 600 ml out) to 7.32% for the lowest volume out (i.e., 0 volume or empty bottle).

On average, the amount of RAE was 3.90%. These results indicate that the trained model can predict larger volume levels with higher accuracy compared to those of lower volumes. The reason is that, as shown in Fig 8 and Fig 9, the accuracy of the ultrasonic sensor may decrease as the volume increases (i.e, the fluid surface is closer to the sensor). Therefore, if we remove those levels which can be measured more accurately while keeping the levels that measured with less accuracy in our train set then our trained model to predict volume level is less accurate. This is because of the limited minimum range of the ultrasonic module. Therefore, this justifies our result that the system measures less accurately while the bottle is almost full, or the fluid surface is close to the sensor.

CHAPTER 5

Components used

- Ultrasonic sensor
- RGB color sensor
- Temperature sensor
- Accelerometer
- Bluetooth Module
- SD card breakout Board

5.1 Waterproof - Ultrasonic sensor A02YYUW

An ultrasonic sensor 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 waves travel faster than the speed of audible sound (i.e. the sound that humans can hear). 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 travelled 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 $D = \frac{1}{2}TxC$ (where D is the distance, T is the time, and C is the speed of sound ~ 343 meters/second). For example, if a scientist set up an ultrasonic sensor aimed at a box and it took 0.025 seconds for the sound to bounce back, the distance between the ultrasonic sensor and the box would be: **$D = 0.5 \times 0.025 \times 343$** or about 4.2875 meters.

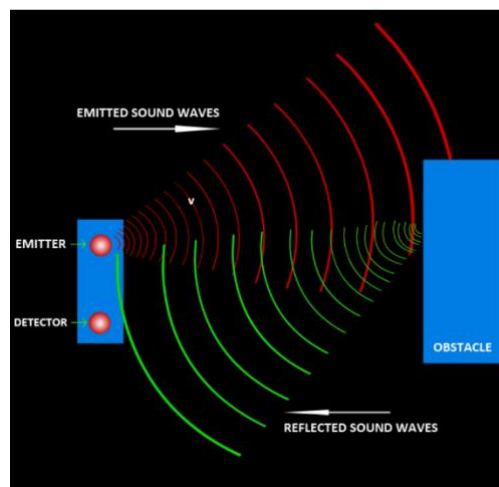


Fig 5.1: Operation of Ultrasonic sensor

Ultrasonic sensors are used primarily as proximity sensors. They can be found in automobile self-parking technology and anti-collision safety systems. Ultrasonic sensors are also used in robotic obstacle detection systems, as well as manufacturing technology. In comparison to infrared (IR) sensors in proximity sensing applications, ultrasonic sensors are not as susceptible to interference of smoke, gas, and other airborne particles (though the physical components are still affected by variables such as heat).

Ultrasonic sensors are also used as level sensors to detect, monitor, and regulate liquid levels in closed containers (such as vats in chemical factories). Most notably, ultrasonic technology has enabled the medical industry to produce images of internal organs, identify tumors, and ensure the health of babies in the womb.

Ultrasonic distance sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse. This is an easy-to-use commercial-grade ultrasonic sensor module of high performance and reliability, featuring much smaller blind zone, wider sensing angle and a certain penetration power(smog, dust) compared with other similar sensors .

The ultrasonic sensor adopts closed separated probe, waterproof and dustproof, which could be well suitable for harsh and moist measuring environment. All the signal processing units are integrated inside the module, so users can directly obtain the distance value through Asynchronous Serial Interface.

A02YYUW is an waterproof ultrasoinic sensor module with 4.5m effective ranging distance. It supports 3.3~5V wide voltage range and is compatible with 3.3V or 5V device like Arduino, Raspberry Pi, etc. The average current of A02YYUW is only 8mA so it can be powered by most controllers' IO port. The ultrasonic sensor adopts closed separated probe, waterproof and dustproof, which could be well suitable for harsh and moist measuring environment. All the signal processing units are integrated inside the module, so users can directly obtain the distance value through Asynchronous Serial Interface. With 9600bit/s band rate, the sensor can easily communicate with upper-host or other MCU, which greatly shortens the developing cycle for users.

SPECIFICATIONS:

- Accuracy: $\pm 1\text{cm}$
- Operating Voltage: 3.3 ~ 5V
- Average Current: $< 8\text{mA}$
- Blind Zone Distance: 3cm
- Detecting Range(Flat object): 3-450cm

- Output: URAT
- Response Time: 100ms
- Operating Temperature: -15~60°C
- Storage Temperature: -25~80°C
- Reference Angle: 60°
- Waterproof Grade: IP67

UART Output

Output Communication

When "RX" floats or input High level, the module outputs processed value, the data is more steady, response time: 100-300ms; when input Low level, the module outputs real-time value, response time: 100ms.

UART	Data bit	Stop bit	Parity	Band rate
TTL level	8	1	none	9600bps

UART Output Form

Frame Data	Description	Byte
Header	0XFF	1 byte
DATA_H	Distance Data High 8-bits	1 byte
DATA_L	Distance Data Low 8-bits	1 byte
SUM	Checksum	1 byte

UART Output

Header	DATA_H	DATA_L	SUM
0xFF	0X07	0XA1	0XA7



Fig 5.2: Waterproof Ultrasonic sensor A02YYUW

5.1.2 Variation of speed of sound

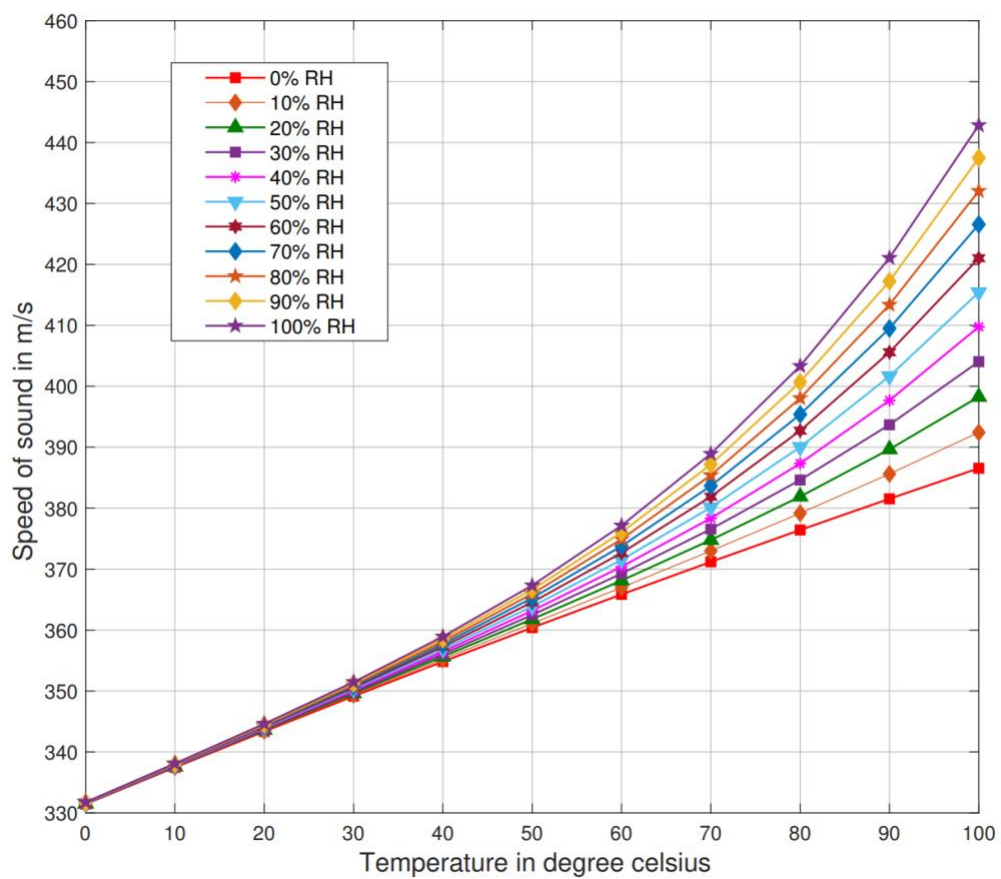


Fig 5.3: The graph to show variation of speed of sound with respect to temperature

5.2 RGB color sensor

What is a color sensor?

Among sensors that detect light, those that detect the three primary colors of red, green, and blue are called color sensors. Color sensors detect RGB values by receiving ambient light using a photodiode.

Color Sensor - Operating Principle

When an object is irradiated with light containing RGB components, the color of the reflected light will change depending on the color of the object.

For example, if the object is red, the reflected light component will be red. For a yellow object, the reflected light will be red and green, and if the object is white all three components will be reflected.

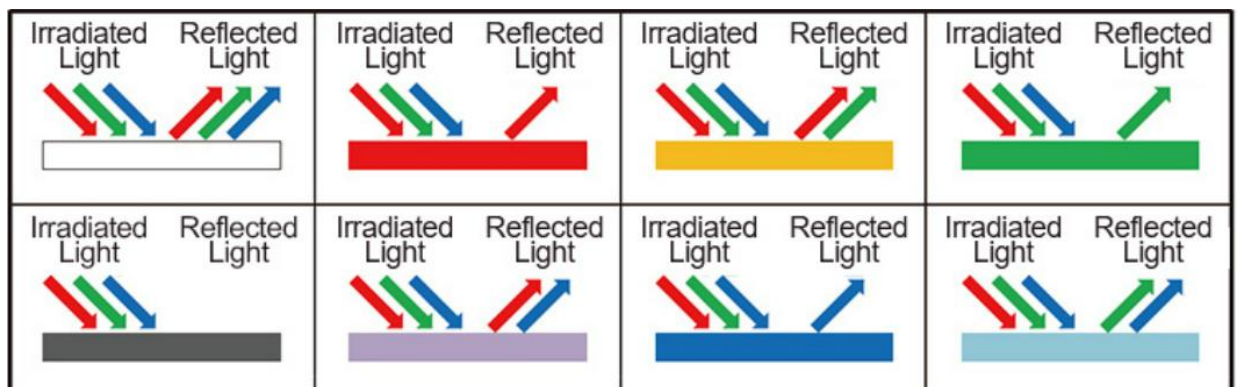


Fig 5.4: Reflected Light for Different Object Colors

In this way, the color of the object is determined from the ratio of color components (RGB) in the reflected light.

Similarly, the human eye determines color by receiving these reflected light components.

It is impossible to see in the dark. This is because there is no irradiating (illuminating) light, which means there is no reflected light so everything looks pitch black.

Like human eyes, color sensors determine the color by first detecting light (using photodiodes) then calculating the ratio of R, G, and B received.

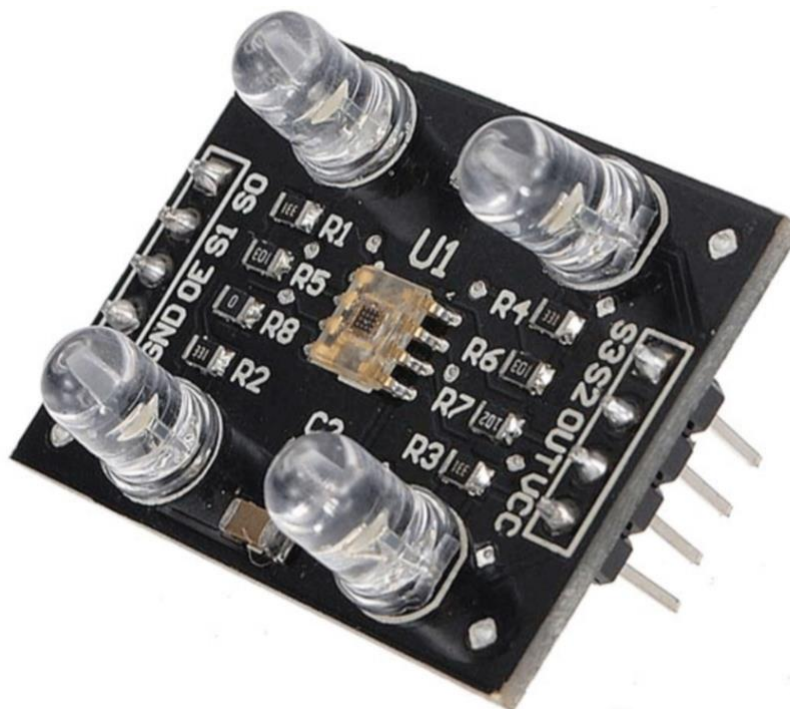


Fig 5.5: TCS3200 RGB sensor

The TCS3200 device provides a digital return of red, green, blue (RGB), and clear light sensing values. An IR blocking filter, integrated on-chip and localized to the color sensing photodiodes, minimizes the IR spectral component of the incoming light and allows color measurements to be made accurately. The high sensitivity, wide dynamic range, and IR blocking filter make the TCS3200 an ideal color sensor solution for use under varying lighting conditions and through attenuating materials. The TCS3200 color sensor has a wide range of applications including RGB LED backlight control, solid-state lighting, health/fitness products, industrial process controls and medical diagnostic equipment. In addition, the IR blocking filter enables the TCS3200 to perform ambient light sensing (ALS). Ambient light sensing is widely used in display-based products such as cell phones, notebooks, and TVs to sense the lighting environment and enable automatic display brightness for optimal viewing and power savings. The TCS3200, itself, can enter a lower-power wait state between light sensing measurements to further reduce the average power consumption.

The TCS3200 light-to-digital converter contains a 3×4 photodiode array, four analog-to-digital converters (ADC) that integrate the photodiode current, data registers, a state machine, and an I2C interface. The 3×4 photodiode array is composed of red-filtered, green-filtered, blue-filtered, and clear (unfiltered) photodiodes. In addition, the photodiodes are coated with an IR-blocking filter. The four integrating ADCs simultaneously convert the amplified photodiode currents to a 16-bit digital value. Upon completion of a conversion

cycle, the results are transferred to the data registers, which are double-buffered to ensure the integrity of the data. All of the internal timing, as well as the low-power wait state, is controlled by the state machine. Communication of the TCS3200 data is accomplished over a fast, up to 400 kHz, two-wire I2C serial bus. The industry standard I2C bus facilitates easy, direct connection to microcontrollers and embedded processors. In addition to the I2C bus, the TCS3200 provides a separate interrupt signal output. When interrupts are enabled, and user-defined thresholds are exceeded, the active-low interrupt is asserted and remains asserted until it is cleared by the controller. This interrupt feature simplifies and improves the efficiency of the system software by eliminating the need to poll the TCS3200. The user can define the upper and lower interrupt thresholds and apply an interrupt persistence filter. The interrupt persistence filter allows the user to define the number of consecutive out-of-threshold events necessary before generating an interrupt. The interrupt output is open-drain, so it can be wire-ORed with other devices.

5.3 DS18B20 Waterproof-Temperature sensor

The DS18B20 waterproof temperature sensor is a pre-wired, one-meter-long, sealed, waterproof digital temperature sensor probe and it is manufactured by Dallas semiconductor and maxim integrated corporations in a waterproof version.

It is a 1-wire digital temperature sensor with 12 bits precision from the onboard digital to analog converter. Its working principle is based on the conversion of temperature to digital format directly and operates in parasite power mode. The operation of this sensor is based on the 1-wire serial communication protocol and stores a 64-bit unique serial code.

As this is a 1-wire digital temperature sensor, it needs only the data pin and GND pin to interface with the Arduino or microcontroller. The temperature sensing of the sensor ranges from -55°C to +125°C with an accuracy of $\pm 5^\circ\text{C}$. It is the best temperature sensor to measure the temperature value at multiple points and requires only one data/digital pin of the Arduino or microcontroller unit to transfer the data.

It operates with 3V to 5.5V positive power supply and consumes a current of a maximum of 1mA. The main advantage of DS18B20 is its alarm function. The output signal can be configured when the values of the temperature reach a high or low threshold value set by the user.

It is a unique 1-wire programmable digital temperature sensor. The operating voltage is 3.0V to 5.5V. It provides an accuracy of $\pm 5^\circ\text{C}$ from -10°C to +85°C. The operating temperature range is -67°F to +257°F or -55°C to +125°C.

Pin Configuration/Pin Out:

The DS18B20 is a 3-pin Or 3-wire waterproof digital temperature sensor. The **DS18B20 waterproof temperature sensor pin configuration/pin diagram** is shown below.



Fig 5.6: DS18B20 Waterproof temperature sensor

Pin 1: GND (Ground): This pin or wire refers to the ground connection of the circuit parasite mode operation. The black wire of the DS18B20 represents the ground pin.

Pin 2: VCC: This pin refers to the positive power supply voltage of 3V to 5.5V to power up the sensor. The red wire of the DS18B20 represents the VCC pin.

Pin 3: DQ or DATA: This pin refers to the output pin that can generate the digital temperature value that can be read by using the 1-wire interface protocol. It is connected to the digital pin of an Arduino or microcontroller while interfacing. It provides power to the sensor when it is operated in the parasitic power mode. The yellow wire of DS18B20 represents the data pin.

Technical Specifications and Features:

The **DS18B20 waterproof temperature technical specifications and features** are given below.

- It is a unique 1-wire programmable digital temperature sensor.
- The operating voltage is 3.0V to 5.5V.
- It provides an accuracy of $\pm 5^{\circ}\text{C}$ from -10°C to $+85^{\circ}\text{C}$.
- The operating temperature range is -67°F to $+257^{\circ}\text{F}$ or -55°C to $+125^{\circ}\text{C}$.
- The selectable resolution is 9 to 12 bits.
- It uses only one digital pin for communication with a 1-wire interface.
- Unique 64-bit ID burned into the chip.
- Multiple temperature sensors can share one pin.
- It is a temperature-limiting alarm system.
- Query time: $<750\text{ms}$.
- It is a 3-wire interface: red wire for VCC, black wire for GND, and yellow wire for DATA.
- It is stainless steel with 6mm diameter and 35mm long.
- Diameter of the cable: 4mm or 0.16.
- Length of the sensor: 95cm Or 37.4".
- Probe: 7mm diameter, 26mm long, 6 feet overall length.
- To prevent short circuits, internal sealing glue, and to provide moisture-proof and waterproof, each pin uses a heat-shrinkable tube.
- It uses a moisture-proof waterproof stainless steel encapsulated tube to prevent rust.
- It uses a unique single bus with a 1-wire serial communication protocol without using any other external components.

5.4 Accelerometer

An accelerometer is an electromechanical device that will measure acceleration force. It shows acceleration, only due to cause of gravity i.e., g force. It measures acceleration in g unit.

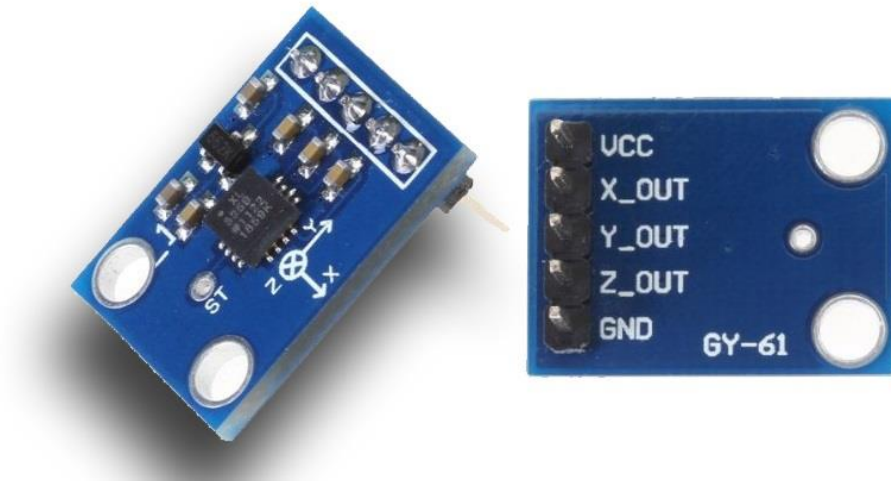


Fig 5.7: ADXL335 Accelerometer

Pin configuration:

VCC: Power supply pin i.e. connect 5V here.

X_OUT: X axis analog output.

Y_OUT: Y axis analog output.

Z_OUT: Z axis analog output.

GND: Ground pin i.e. connect ground here.

ADXL335 accelerometer provides analog voltage at the output X, Y, Z pins; which is proportional to the acceleration in respective directions i.e. X, Y, Z.

On the earth, 1g means acceleration of 9.8 m/s^2 is present. On moon, it is 1/6th of earth and on mars it is 1/3rd of earth.

Accelerometer can be used for tilt-sensing applications as well as dynamic acceleration resulting from motion, shock, or vibration.

- The ADXL335 gives complete 3-axis acceleration measurement.
- This module measures acceleration within range $\pm 3 \text{ g}$ in the x, y and z axis.
- The output signals of this module are analog voltages that are proportional to the acceleration.
- It contains a polysilicon surface-micro machined sensor and signal conditioning circuitry.

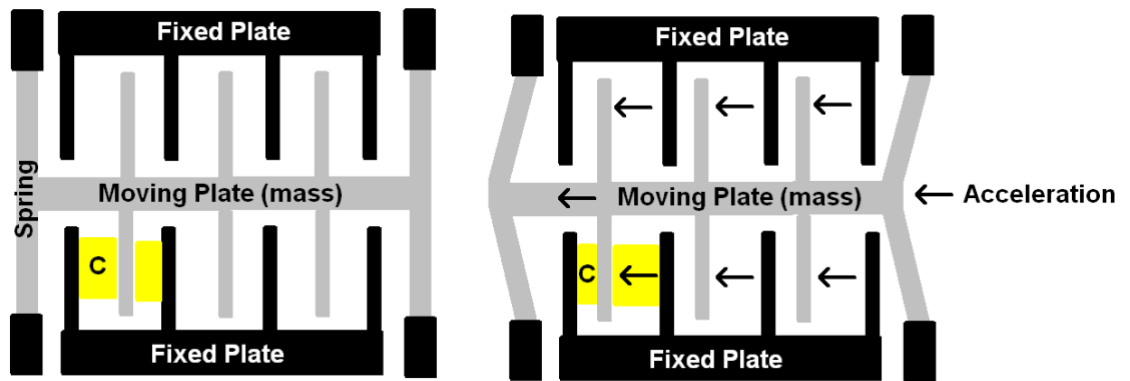


Fig 5.8: Accelerometer sensor MEM Mechanism

- As we can see from the above figure, basic structure of accelerometer consists fixed plates and moving plates (mass).
- Acceleration deflects the moving mass and unbalances the differential capacitor which results in a sensor output voltage amplitude which is proportional to the acceleration.
- Phase-sensitive demodulation techniques are then used to determine the magnitude and direction of the acceleration.

Angles using ADXL335

We can calculate angle of inclination or tilt by using X, Y, Z's value. Also, we can calculate Roll, Pitch and Yaw angles with respect to X, Y and Z axis. So first we need to convert 10-bit ADC values into g unit.

As per ADXL335 datasheet maximum voltage level at 0g is 1.65V and sensitivity scale factor of 330mV/g.

$$A_{out} = \frac{\frac{ADC \text{ value} * V_{ref}}{1024} - \text{Voltage Level at } 0g}{\text{Sensitivity Scale factor}} \quad \dots \dots (5.1)$$

Above formula gives us acceleration values in g unit for X, Y and Z axis as,

$$\mathbf{Axout} = (((X \text{ axis ADC value} * V_{ref}) / 1024) - 1.65) / 0.330$$

$$\mathbf{Ayout} = (((Y \text{ axis ADC value} * V_{ref}) / 1024) - 1.65) / 0.330$$

$$\mathbf{Azout} = (((Z \text{ axis ADC value} * V_{ref}) / 1024) - 1.65) / 0.330$$

Note that, practically we get slightly different voltage at 0g. So, put the practical value of voltage at 0g.

5.5 Bluetooth Module HC-05

HC-05 Bluetooth Module is an easy-to-use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Its communication is via serial communication which makes an easy way to interface with controller or PC. HC-05 Bluetooth module provides switching mode between master and slave mode which means it able to use neither receiving nor transmitting data.

Specification:

- Model: HC-05
- Input Voltage: DC 5V
- Communication Method: Serial Communication
- Master and slave mode can be switched
- It is used for many applications like wireless headset, game controllers, wireless mouse, wireless keyboard and many more consumer applications.
- It has range up to <100m which depends upon transmitter and receiver, atmosphere, geographic & urban conditions.
- It is IEEE 802.15.1 standardized protocol, through which one can build wireless Personal Area Network (PAN). It uses frequency-hopping spread spectrum (FHSS) radio technology to send data over air.
- It uses serial communication to communicate with devices. It communicates with microcontroller using serial port (USART).

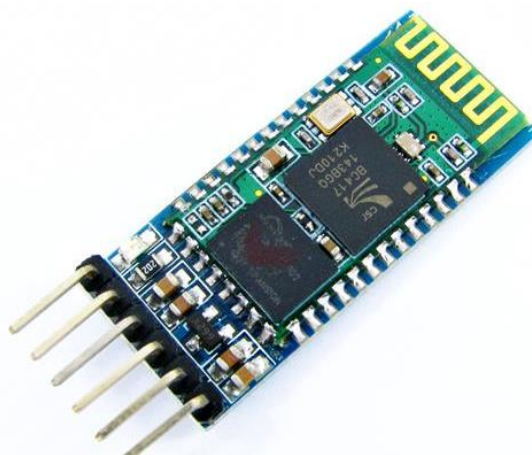


Fig 5.9: Bluetooth Module HC-05

Pin Description:



Fig 5.10: Pin description of HC-05 Bluetooth module

Bluetooth serial modules allow all serial enabled devices to communicate with each other using Bluetooth.

It has 6 pins,

1. **Key/EN:** It is used to bring Bluetooth module in AT commands mode. If Key/EN pin is set to high, then this module will work in command mode. Otherwise by default it is in data mode. The default baud rate of HC-05 in command mode is 38400bps and 9600 in data mode.

HC-05 module has two modes,

1. **Data mode:** Exchange of data between devices.
 2. **Command mode:** It uses AT commands which are used to change setting of HC-05. To send these commands to module serial (USART) port is used.
2. **VCC:** Connect 5 V or 3.3 V to this Pin.
 3. **GND:** Ground Pin of module.
 4. **TXD:** Transmit Serial data (wirelessly received data by Bluetooth module transmitted out serially on TXD pin)
 5. **RXD:** Receive data serially (received data will be transmitted wirelessly by Bluetooth module).
 6. **State:** It tells whether module is connected or not.

5.6 SD card breakout Board

Not just a simple breakout board, this microSD adapter goes the extra mile - designed for ease of use.

- Onboard 5v->3v regulator provides 150mA for power-hungry cards
- 3v level shifting means you can use this with ease on either 3v or 5v systems
- Uses a proper level shifting chip, not resistors: less problems, and faster read/write access
- Use 3 or 4 digital pins to read and write 2Gb+ of storage!
- Activity LED lights up when the SD card is being read or written
- Four #2 mounting holes
- Push-push socket with card slightly over the edge of the PCB so its easy to insert and remove
- Comes with 0.1" header (unattached) so you can get it on a breadboard or use wires - your choice
- Tested and assembled here at the Adafruit factory
- Works great with Arduino, with tons of example code and wiring diagrams

To use with an Arduino, connect GND to ground, 5V to 5V, CLK to pin 13, DO to pin 12, DI to pin 11, and CS to pin 10. Then you can use the Arduino IDE's SD library which supports FAT and FAT32 SD cards.

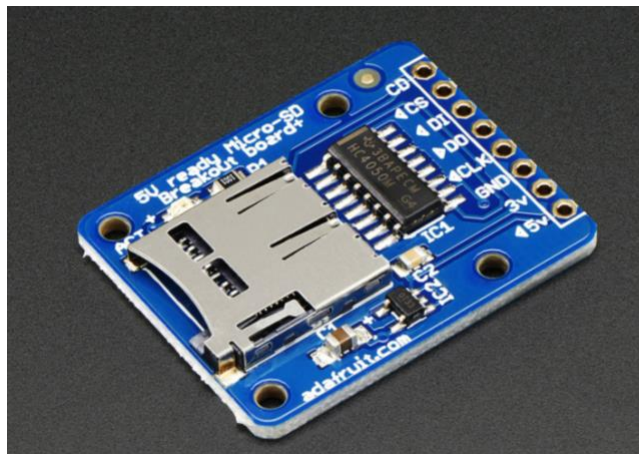


Fig 5.11: SD card breakout board

5.7 Wi-Fi Module Esp8266-01

The **ESP8266 ESP-01** is a Wi-Fi module that allows **microcontrollers** access to a **Wi-Fi network**. This module is a self-contained **SOC** (System On a Chip) that doesn't necessarily need a microcontroller to manipulate inputs and outputs as you would normally do with an **Arduino**, for example, because the ESP-01 acts as a small computer. Depending on the version of the ESP8266, it is possible to have up to 9 GPIOs (General Purpose Input Output). Thus, we can give a microcontroller internet access like the Wi-Fi shield does to the Arduino, or we can simply program the ESP8266 to not only have access to a Wi-Fi network, but to act as a microcontroller as well. This makes the ESP8266 very versatile, and it can save you some money and space in your projects.

KEY FEATURES OF ESP8266 ESP-01 WIFI MODULE:

- 32-bit RISC Tensilica Xtensa LX Processor running at 80MHz
- 1MB Flash Memory
- IEEE 802.11 b/g/n Wi-Fi
- 2 GPIO
- 3.3V Operation

The module requires 3.3V for power and is not 5V tolerant on its pins. If connecting to a 5V MCU, the RX pin of the ESP-01 module which is driven by the TX pin from the MCU should have a level shifter installed. This can just be a simple voltage divider using 2 resistors as shown down below. If you are using it with our ESP-01 adapter, the adapter includes level shifting as well as a 3.3V regulator, so it makes the ESP-01 module plug-n-play.

The module has a red power LED and a blue LED to indicate data flow. The WiFi antenna is the meandering PCB trace that covers the end of the module.

The ESP8266 in the name refers to the 32-pin Espressif IC on the module. These chips are quite capable and support up to 16 GPIO. The ESP-01 is the most basic and least expensive implementation of this chip with limited GPIO and is best used for adding WiFi to an existing Arduino or similar MCU board. If you are looking to use it as a stand-alone device, you will typically be better off starting with something like the NodeMCU or ESP32 that we sell which includes all the features of a full microcontroller like the Arduino and can be programmed directly from the Arduino IDE.

From the factory, the modules are loaded with "AT" firmware and can be programmed via a simple terminal program. If you are using the module primarily to exploit its WiFi

capabilities and are controlling it with another MCU like an Arduino, that may be all that you need.

If using with custom software programmed onto the device, it can be programmed using the Arduino IDE but it will require the use of a FTDI USB to Serial interface.

The module has a 2×4-pin header on the assembly. See pic for layout as the boards may not have the pins labeled.

2 x 4 Header:

- **GND** = Ground. Connect to ground on MCU
- **GPIO2** = General Purpose Digital I/O
- **GPIO0** = General Purpose Digital I/O
- **RX** = Receive Data. Connects to RX on MCU
- **TX** = Transmit Data. Connects to TX on MCU
- **CH_PD** = Enable / Power Down. Must be pulled to 3.3V directly or via pull-up resistor to enable
- **RST** = Reset. Active low, must be pulled to 3.3V directly or via pull-up resistor
- **VCC** = 3.3V. Can draw up to 200mA worse case.

The ESP8266 ESP-01 module has three operation modes:

1. **Access Point (AP)**
2. **Station (STA)**
3. **Both**

In **AP** the Wi-Fi module acts as a Wi-Fi network, or access point (hence the name), allowing other devices to connect to it. This does not mean that you will be able to check your Facebook from your device while the ESP-01 module is operating in the AP mode. It simply establishes a two way communication between the ESP8266 and the device that is connected to it via Wi-Fi.

In **STA** mode, the ESP-01 can connect to an AP such as the Wi-Fi network from your house. This allows any device connected to that network to communicate with the module.

The third mode of operation permits the module to act as both an AP and a STA.

The corresponding number for each mode of operation is as follows:

- STA = 1
- AP = 2
- Both = 3

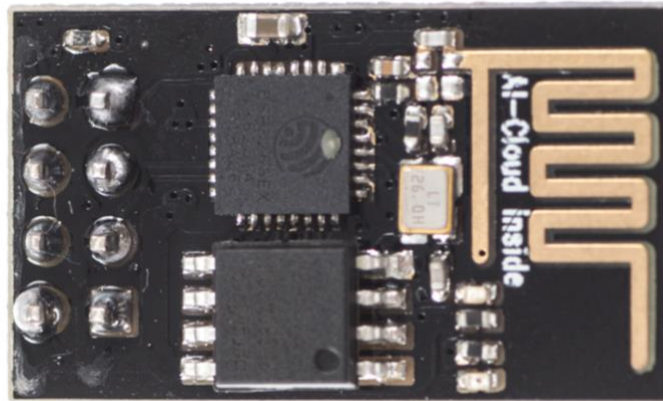


Fig 5.12: Esp8266-01 WiFi Module

CHAPTER 6

EMBEDDED SYSTEM

6.0 Introduction to embedded system

An embedded system is a combination of computer hardware and software designed for a specific function. Embedded systems may also function within a larger system. The systems can be programmable or have a fixed functionality. Industrial machines, consumer electronics, agricultural and processing industry devices, automobiles, medical equipment, cameras, digital watches, household appliances, airplanes, vending machines and toys, as well as mobile devices, are possible locations for an embedded system.

While embedded systems are computing systems, they can range from having no user interface (UI) -- for example, on devices designed to perform a single task -- to complex graphical user interfaces (GUIs), such as in mobile devices. User interfaces can include buttons, LEDs (light-emitting diodes) and touchscreen sensing. Some systems use remote user interfaces as well.

MarketsandMarkets, a business-to-business (B2B) research firm, predicted that the embedded market will be worth \$116.2 billion by 2025. Chip manufacturers for embedded systems include many well-known technology companies, such as Apple, IBM, Intel and Texas Instruments. The expected growth is partially due to the continued investment in artificial intelligence (AI), mobile computing and the need for chips designed for high-level processing.

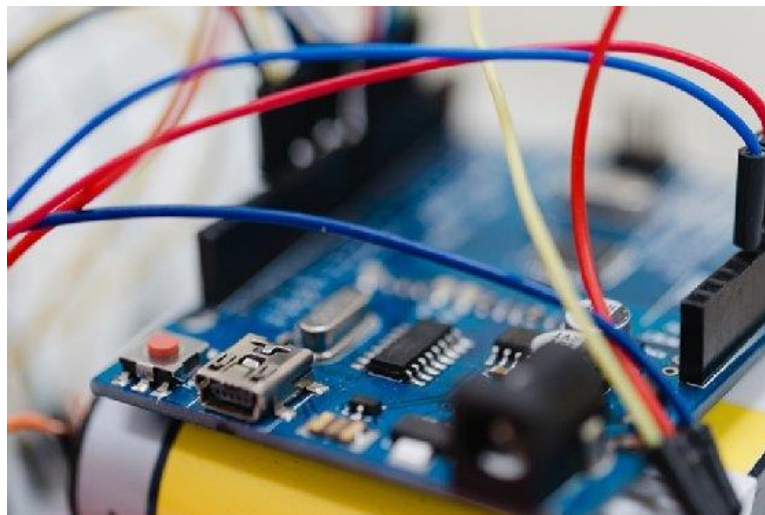


Fig 6.1: Arduino Microcontroller connected with wires

6.1 Examples of embedded systems

Embedded systems are used in a wide range of technologies across an array of industries.

Some examples include:

- **Automobiles.** Modern cars commonly consist of many computers (sometimes as many as 100), or embedded systems, designed to perform different tasks within the vehicle. Some of these systems perform basic utility functions and others provide entertainment or user-facing functions. Some embedded systems in consumer vehicles include cruise control, backup sensors, suspension control, navigation systems and airbag systems.
- **Mobile phones.** These consist of many embedded systems, including GUI software and hardware, operating systems (OSes), cameras, microphones, and USB (Universal Serial Bus) I/O (input/output) modules.
- **Industrial machines.** They can contain embedded systems, like sensors, and can be embedded systems themselves. Industrial machines often have embedded automation systems that perform specific monitoring and control functions.
- **Medical equipment.** These may contain embedded systems like sensors and control mechanisms. Medical equipment, such as industrial machines, also must be very user-friendly so that human health isn't jeopardized by preventable machine mistakes. This means they'll often include a more complex OS and GUI designed for an appropriate UI.

6.2 How does an embedded system work?

Embedded systems always function as part of a complete device -- that's what's meant by the term *embedded*. They are low-cost, low-power-consuming, small computers that are embedded in other mechanical or electrical systems. Generally, they comprise a processor, power supply, and memory and communication ports. Embedded systems use the communication ports to transmit data between the processor and peripheral devices -- often, other embedded systems -- using a communication protocol. The processor interprets this data with the help of minimal software stored on the memory. The software is usually highly specific to the function that the embedded system serves.

The processor may be a microprocessor or microcontroller. Microcontrollers are simply microprocessors with peripheral interfaces and integrated memory included. Microprocessors use separate integrated circuits for memory and peripherals instead of including them on the

chip. Both can be used, but microprocessors typically require more support circuitry than microcontrollers because there is less integrated into the microprocessor. The term *system on a chip (SoC)* is often used. SoCs include multiple processors and interfaces on a single chip. They are often used for high-volume embedded systems. Some example SoC types are the application-specific integrated circuit (ASIC) and the field-programmable gate array (FPGA).

Often, embedded systems are used in real-time operating environments and use a real-time operating system (RTOS) to communicate with the hardware. Near-real-time approaches are suitable at higher levels of chip capability, defined by designers who have increasingly decided the systems are generally fast enough and the tasks tolerant of slight variations in reaction. In these instances, stripped-down versions of the Linux operating system are commonly deployed, although other OSes have been pared down to run on embedded systems, including Embedded Java and Windows IoT (formerly Windows Embedded).

6.3 Characteristics of embedded systems

The main characteristic of embedded systems is that they are task-specific.

Additionally, embedded systems can include the following characteristics:

- typically, consist of hardware, software and firmware;
- can be embedded in a larger system to perform a specific function, as they are built for specialized tasks within the system, not various tasks;
- can be either microprocessor-based or microcontroller-based -- both are integrated circuits that give the system compute power;
- are often used for sensing and real-time computing in internet of things (IoT) devices, which are devices that are internet-connected and do not require a user to operate;
- can vary in complexity and in function, which affects the type of software, firmware and hardware they use; and
- are often required to perform their function under a time constraint to keep the larger system functioning properly.

6.4 Structure of embedded systems

Embedded systems vary in complexity but, generally, consist of three main elements:

- **Hardware.** The hardware of embedded systems is based around microprocessors and microcontrollers. Microprocessors are very similar to microcontrollers and, typically,

refer to a CPU (central processing unit) that is integrated with other basic computing components such as memory chips and digital signal processors (DSPs). Microcontrollers have those components built into one chip.

- **Software and firmware.** Software for embedded systems can vary in complexity. However, industrial-grade microcontrollers and embedded IoT systems usually run very simple software that requires little memory.
- **Real-time operating system.** These are not always included in embedded systems, especially smaller-scale systems. RTOSes define how the system works by supervising the software and setting rules during program execution.

In terms of hardware, a basic embedded system would consist of the following elements:

- **Sensors** convert physical sense data into an electrical signal.
- **Analog-to-digital (A-D) converters** change an analog electrical signal into a digital one.
- **Processors** process digital signals and store them in memory.
- **Digital-to-analog (D-A) converters** change the digital data from the processor into analog data.
- **Actuators** compare actual output to memory-stored output and choose the correct one.

The sensor reads external inputs, the converters make that input readable to the processor, and the processor turns that information into useful output for the embedded system.

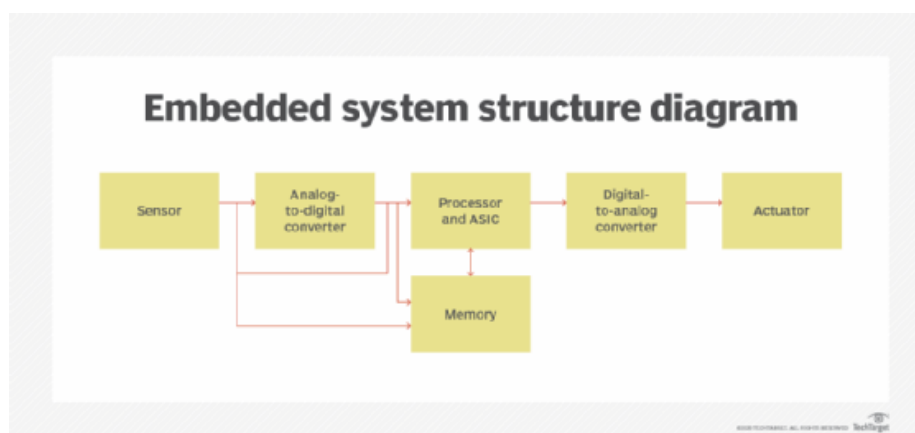


Fig 6.2: Structure of Embedded system

CHAPTER 7

SOFTWARE

7.0 Introduction of software modules:

In order to work with this project, first the right of software implementation needs to be identified. The preliminary works for software searching are:

- i. Learn the usage of Embedded C language and its criteria based on the project objectives.
- ii. Identify the memory usage required by the program that will be built whether it suits with the microcontroller or not.
- iii. Identify the Software that will be used to load the program to the chip using boot loader techniques
- iv. Check and explore the entire menu in the software used in order to achieve the project objectives.

7.1 ARDUINO IDE:

The Arduino Integrated Development Environment – or Arduino Software (IDE) – contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate them.

Writing Sketches:

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino IDE, including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor. NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension .pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.

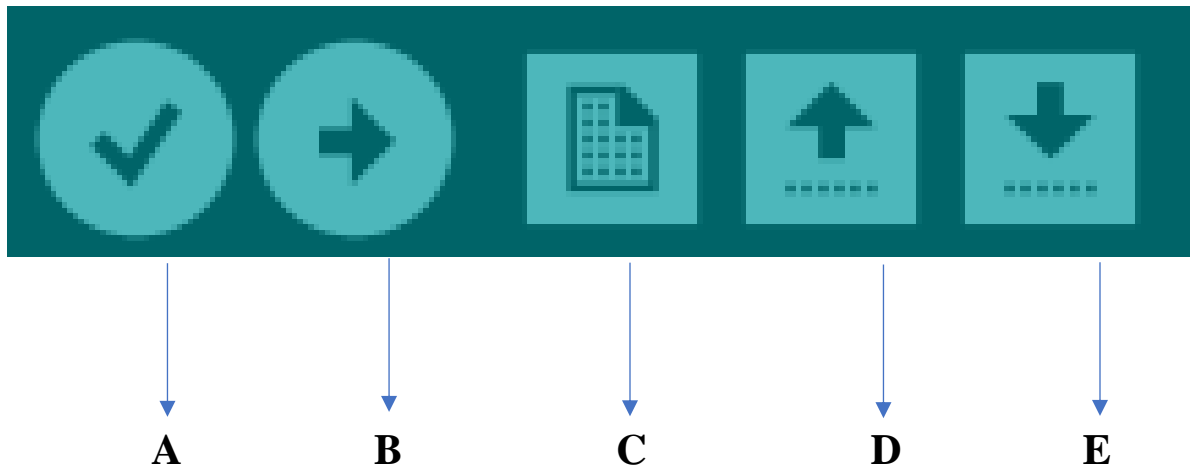


Fig 7.1: Tool bar of Arduino IDE

A. *Verify*

Checks your code for errors. (Compilation).

B. *Upload*

Compiles your code and uploads it to the configured board.

Note: If you are using an external programmer with your board, you can hold down the “shift” key on your computer when using this icon. The text will change to “Upload using Programmer”

C. *New*

Shortcut used to create a new sketch

D. *Open*

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Note: due to a bug in java, this menu doesn’t scroll, if you need to open a sketch late in the list, use the File | Sketch book menu instead

E. *Save*

Saves your sketch

Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

7.1.1 Arduino IDE Initial setup:

This is the Arduino IDE once it’s been opened. It opens into a blank sketch where you can start programming immediately. First, we should configure the board and port settings to allow us to upload code. Connect your Arduino board to the PC via USB cable.

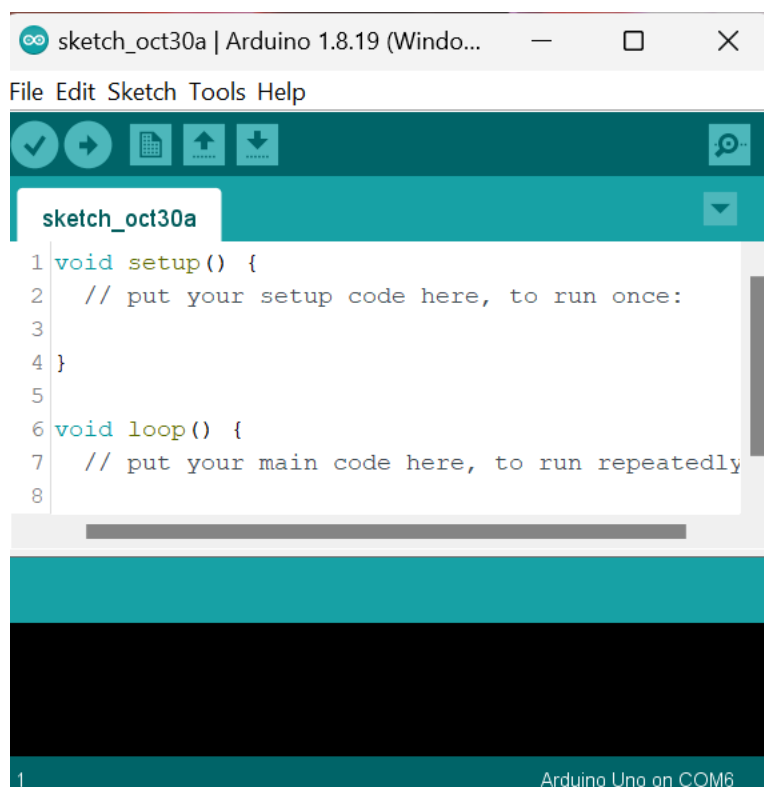


Fig 7.2: Arduino IDE window

7.1.2 IDE: Board Setup

You have to tell the Arduino IDE what board you are uploading to. Select the Tools pulldown menu and go to Board. This list is populated by default with the currently available Arduino Boards that are developed by Arduino. If you are using an Uno or an Uno-Compatible Clone (ex. Funduino, SainSmart, IEIK, etc.), select Arduino Uno. If you are using another board/clone, select that board.

7.1.3 IDE: COM Port Setup

If you downloaded the Arduino IDE before plugging in your Arduino board, when you plugged in the board, the USB drivers should have installed automatically. The most recent Arduino IDE should recognize connected boards and label them with which COM port they are using. Select the Tools pulldown menu and then Port. Here it should list all open COM ports, and if there is a recognized Arduino Board, it will also give it's name. Select the Arduino board that you have connected to the PC. If the setup was successful, in the bottom right of the Arduino IDE, you should see the board type and COM number of the board you plan to

program. Note: the Arduino Uno occupies the next available COM port; it will not always be COM3.

7.1.4 Testing Your Settings:

Uploading Blink One common procedure to test whether the board you are using is properly set up is to upload the "Blink" sketch. This sketch is included with all Arduino IDE releases and can be accessed by the Filepull-down menu and going to Examples, 01.Basics, and then select Blink. Standard Arduino Boards include a surface-mounted LED labeled "L" or "LED" next to the "RX" and "TX" LEDs, that is connected to digital pin 13. This sketch will blink the LED at a regular interval, and is an easy way to confirm if your board is set up properly and you were successful in uploading code. Open the "Blink" sketch and press the "Upload" button in the upper-left corner to upload.

7.2 Burning Boot loader of Arduino board

The bootloader is a small piece of software that allows uploading of sketches onto the Arduino board. It comes preprogrammed on the microcontrollers on Arduino boards. Whether the bootloader has been corrupted or intentionally has been removed, it can be restored by *burning* (also called, *flashing* or *programming*) a new bootloader to the board.

The easiest way to burn the bootloader to classic AVR boards (UNO, Mega, Nano, etc.) is using **a second Arduino board** as a programmer, which is the method that will be covered below.

Step 1: Connect the boards

AVR boards are programmed with the SPI interface (COPI, CIPO and SCK signals). On many boards (including UNO, Mega, and Nano), these are available in two locations:

The digital and power pins. These are the most commonly used pins on Arduino, and you'll only need six standard male-to-male jumper wires.

The ICSP header. CIPO, COPI, and SCK are available in a consistent physical location on the ICSP header. You'll need female-to-female jumper cables for these pins.

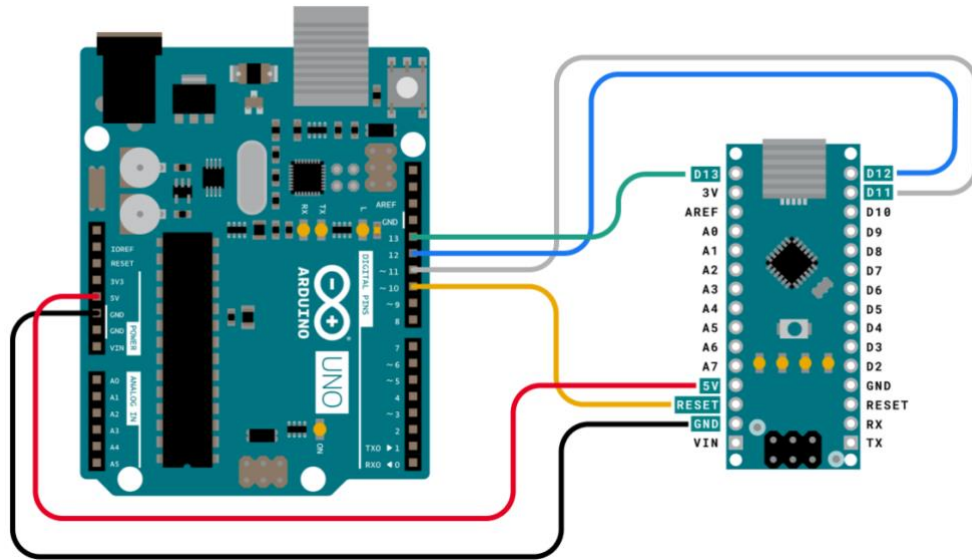


Fig 7.3: Connections for burning the boot loader of Arduino Nano using an Arduino Uno

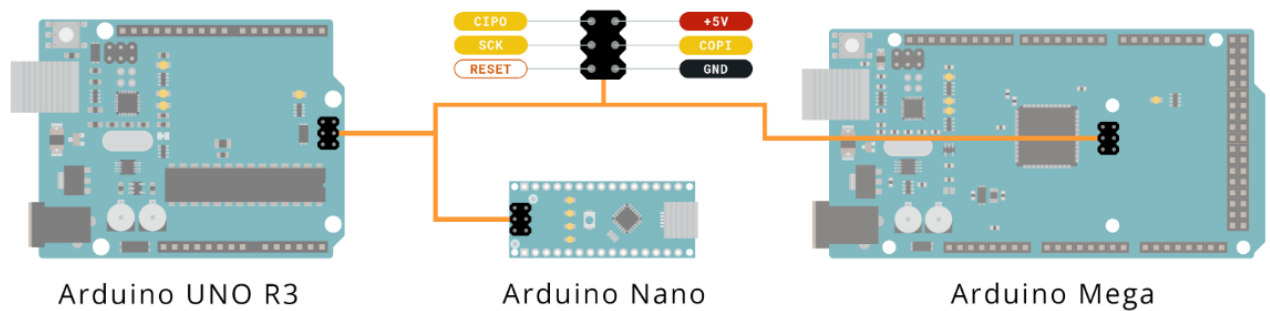


Fig 7.4: Connections for burning the boot loader, Using ICSP Header

Step 2: Burn the bootloader in Arduino IDE

Now that the board connections are set up, we can use Arduino IDE to burn the bootloader.

1. Open Arduino IDE.
2. Connect the **programmer board** to the computer.
3. Select your programmer board in *Tools > Board*, for example *Tools > Board > Arduino AVR Boards > Arduino Uno*.
4. Select the port the board is connecting to in *Tools > Port*.
5. Open *File > Examples > 11.ArduinoISP > ArduinoISP*.

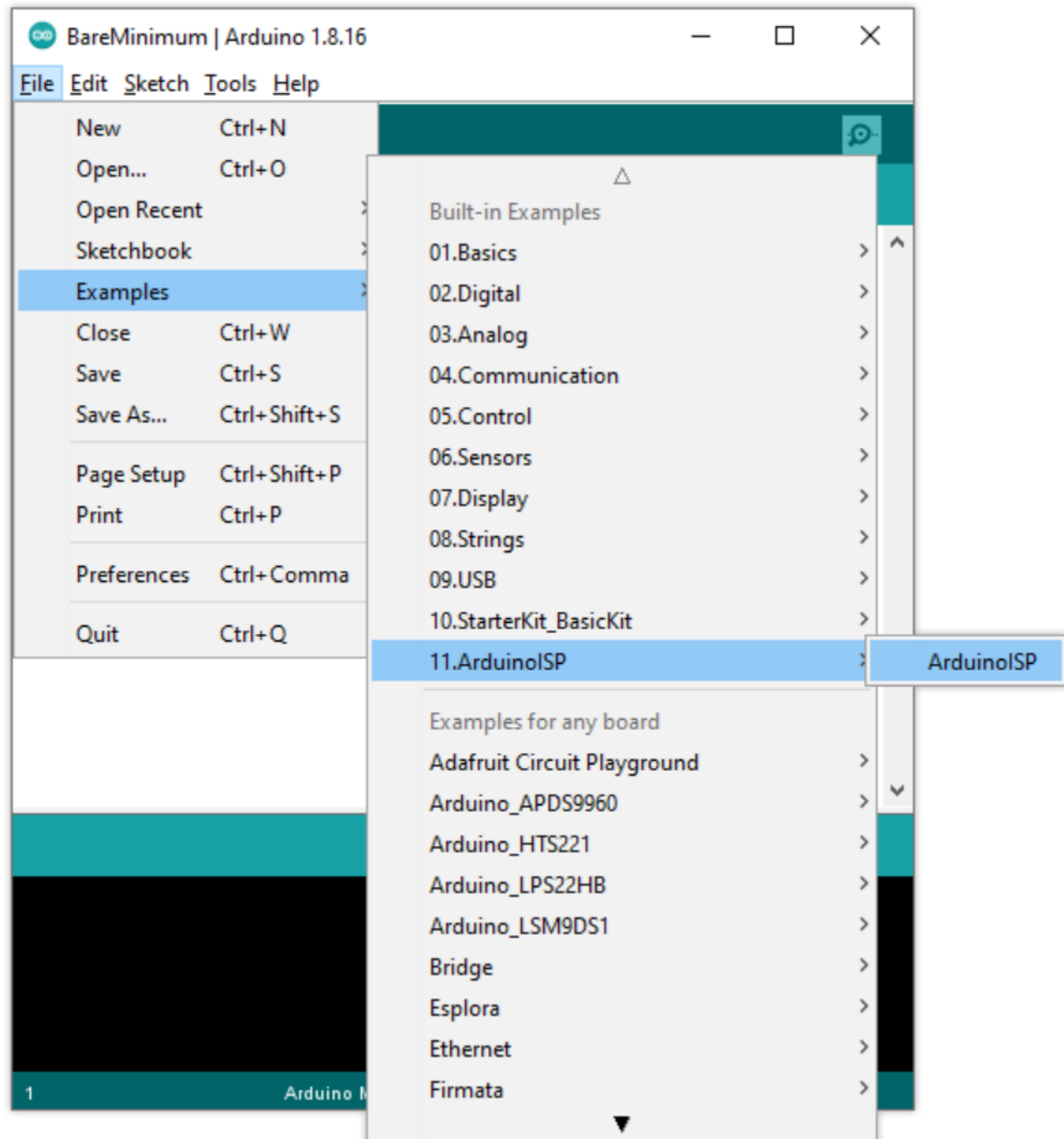



Fig 7.5: Selecting the Programmer options

6. Click  **Upload** to upload the sketch to the board.
7. Select the **target board** in *Tools > Board*.
8. Select *Tools > Programmer > Arduino as ISP*.
9. We can now start the burning process. It's a good idea to enable [verbose upload output](#) in preferences since there will be no console output during the process otherwise. To start the process, select *Tools > Burn Bootloader*.

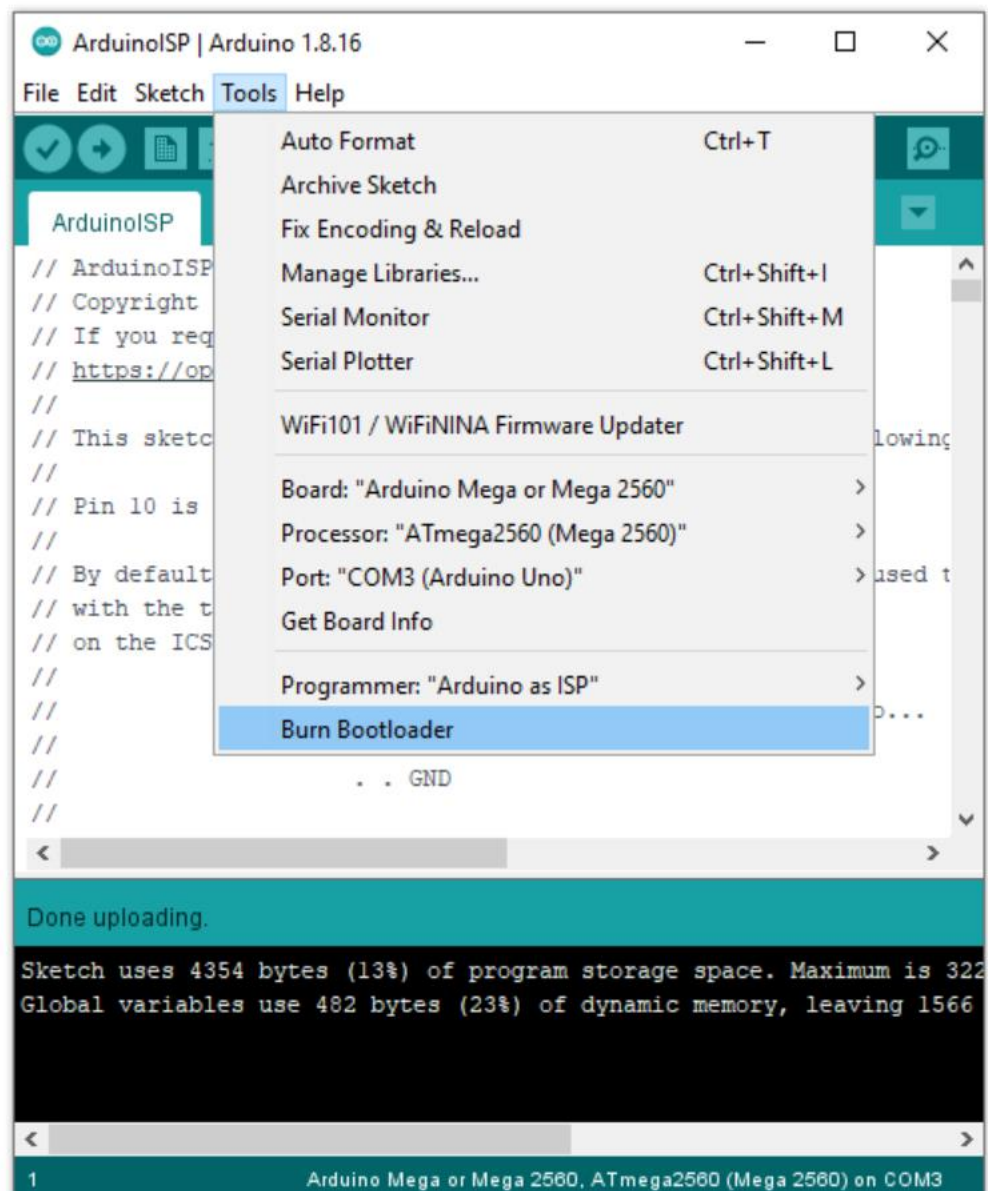


Fig 7.6: Burning Boot loader

Wait for the process to complete, which may take up to a minute. When it's finished, the message "Done burning bootloader" will appear on top of the console, or in a bottom-right notification pop-up, depending on your IDE version.



Fig 7.7: Acknowledgement after successful burning

CHAPTER 8

RESULTS AND CONCLUSION

Conclusion :

We designed, developed, and evaluated a low-power and portable sensor system for detecting fluid type and volume using low-cost, commercially available sensors including accelerometer, RGB, ultrasonic and temperature sensors which cost less than \$50. We have implemented feature engineering and machine learning algorithms to process raw sensor data to detect fluid volume, fluid type and bottle state. Using Random Forest Regressor and Feature Embedding method, the system can estimate the fluid volume with a RAE of 1.1% while detecting the fluid type of 10 different fluids with an accuracy of 97.6% for previously seen bottle. In addition, fluid type detection accuracy for unseen bottles using feature embedding method obtained the average of 84.8%. LIDS is also able to identify the bottle state as expressed by movements of the lid with an accuracy of 98%. In this paper, we provided the results of fluid type classification, fluid volume estimation, drinking event recognition, consumed calorie estimation through fluids and current draw of the system.

We conducted a comprehensive set of experiments to evaluate the performance of LIDS. The analysis focused on performance evaluation of various machine learning algorithms used for fluid volume estimation, type detection, and bottlestate recognition. In addition, we experimentally measured the power consumption of the system for various combinations of the sensors that can be included in the prototype system. We installed LIDS on commercially available water bottles for validation.

Accuracy Results:

For color accuracy:

Table 8.1: Accuracy based on different Illuminance conditions

Light setting	Illuminance	Accuracy (%)
Dark	0	93.81
Medium	200	96.61
Extreme	1000	98.81

Absolute relative errors in LOVO approach:

Table 8.2: Relative absolute error approximated based on inputs

Volume range	Test set(mL)	Relative absolute error(%)
150,300,450,600	0	7.32
0,300,450,600	150	4.16
0,150,450,600	300	3.80
0,150,300,600	450	2.73
0,150,300,450	600	1.52

CHAPTER 9

FUTURE SCOPE

9.1 Future scope

As discussed, LIDS measures the fluid volume based on the distance from the sensor to fluid surface. We tested this method by using multiple cylinder shape bottles with fixed radii. Therefore, we estimate volume by having the height of the fluid and bottle radius which is given through calibration phase. However, there are bottles or cups in the market with variable radii. For instance, some bottles have smaller radius near their spouts and larger radius at the bottom of the bottles. For measuring the fluid volume in these bottles, we may extend the calibration phase by asking the user to pour 4 equal volumes (0%, 25%, 50%, 75%) of fluid into the bottle in calibration phase. By having these levels, we can train a regression model which can estimate fluid volume of the new bottle. We conclude that we need the information regarding the bottle shape to measure fluid volume which can be achieved through one or more calibration process depending on the bottle shape. Then these settings can be saved in the mobile app for any new bottles that users wish to use with LIDS. We trained different regression (volume estimation function) and classification (bottle-state recognition and type detection functions) models in this work.

We trained popular and widely used machine learning models for volume estimation and type detection. We compare the results of the tested classifiers and choose the one with the best performance on the collected data. LIDS requires a fast prediction model for bottle state recognition because it is the only function that runs continuously, and it activates other two functions. As a result, we need a simple and fast model that can be implemented on the microcontroller. Therefore, we train a decision tree model, which can be implemented using few if/then/else statements. Extracted features are passed through this simple model for continuous bottle-state recognition. Fluid type detection of an unseen bottle proved to be challenging based on the results of the LOBO experiment.

As it can be seen, the train and test data are not from the same distribution while testing an unseen bottle color. Therefore, we need to address out-of-distribution generalization problem. To this end, we utilize the feature embedding method by training a neural network for type detection to find a shared color space between all bottles. However, we do not face the same problem for volume estimation and bottle-state recognition. As a result, LIDS only uses traditional machine learning models for bottlestate recognition and volume estimation functions. Considering the 500 mAh battery, LIDS can operate for

approximately 42.3 hours. We recognize that this duration can be improved by optimizing various sources of power consumption.

Therefore, we reduced the current draw of the system by reducing the clock speed of the microcontroller from 16 MHz to 8 MHz. The power consumption of bottle-state recognition, type detection, consumed volume detection states reported as 32.3 mW, 88.9 mW, and 62.8 mW, respectively. This reduces the overall power consumption of LIDS by 17.8 mW and increases the battery lifetime to 60.6 hours. Another possible approach to reduce the power consumption is to remove the ultrasonic sensor, which is the main contributor to the power consumption. As interesting research question is whether or not the RGB color sensor can be used to measure the fluid volume.

A preliminary analysis on training a regression model using 1) ultrasonic, RGB color and temperature sensors; and 2) using RGB color and temperature sensors provided the following results. The RRSE of fluid volume estimation while feeding all three sensors' data was 1.12% as discussed in our results section. However, the RRSE value while using only the RGB color and temperature sensors was 15.63%. The potential of RGB sensor to estimate fluid volume can be explained by the change in light reflections as a result of changes in the fluid volume. We plan to investigate this approach in more details in the future which can reduce the power consumption. As part of our future work, we also plan to incorporate a beverage hydration index (BHI) based on the outputs from LIDS.

A beverage hydration index will indicate the rate of fluid absorption of each fluid and helps determine the most effective fluid volume for different types of beverages. Furthermore, part of our future work will also involve tracking daily activities of user to provide more accurate feedback based on the intensity of activities and hydration requirements. Moreover, we also plan on monitoring real-time hydration status from skin using non-invasive sensors. Fusing various data including ambient temperature, activity level, and hydration status will result in a system that can dynamically provide feedback with enhanced system performance.

9.2 Extension Of This Application:

Infusion device is a means for injecting certain chemical fluids, nutritional fluids, blood transfusions and chemotherapy to patients. The fluid enters the body through veins. The use of infusion is actually not so problematic if the patient can be controlled and monitored periodically for a short time by the nurse. Some problems present when there is lack of human resources in the hospital or nurse's negligence. One of the problems is the

administration of intravenous fluids. When the fluid runs out, there is no sign or warning directly sent to the nurse.

Therefore, a system to control and monitor the level of infusion fluid is necessary to design. The system is designed to detect the level of infusion fluid using a level sensor and a microcontroller as the data processor and hardware regulator. This system sends messages to nurses by short message text application.

The result of the research showed that when the infusion fluid is about to finish, the system notifies the nurse by sending an SMS.

CHAPTER 10

ARDUINO SKETCH

```
#include <SoftwareSerial.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 2
#define S0 4
#define S1 5
#define S2 6
#define S3 7
#define sensorOut 8

OneWire oneWire(ONE_WIRE_BUS);

DallasTemperature sensors(&oneWire);

SoftwareSerial mySerial(11,10); // RX, TX

unsigned char data[4]={ };

float distance,d,tempc;
float x, y, z;

int r,g,b;
int frequency = 0;

void setup()
{
    pinMode(S0, OUTPUT);
    pinMode(S1, OUTPUT);
    pinMode(S2, OUTPUT);
    pinMode(S3, OUTPUT);
    pinMode(sensorOut, INPUT);

    // Setting frequency-scaling to 20%
    digitalWrite(S0,HIGH);
    digitalWrite(S1,LOW);

    Serial.begin(9600);

    mySerial.begin(9600);

    sensors.begin();

    sensors.requestTemperatures();
    // Setting red filtered photodiodes to be read
    digitalWrite(S2,LOW);
```

```
digitalWrite(S3,LOW);
// Reading the output frequency
frequency = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255
frequency = map(frequency, 25,72,255,0);

// Printing the value on the serial monitor
r = frequency;
delay(100);

// Setting Green filtered photodiodes to be read
digitalWrite(S2,HIGH);
digitalWrite(S3,HIGH);

// Reading the output frequency
frequency = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255
frequency = map(frequency, 30,90,255,0);

// Printing the value on the serial monitor
g = frequency;
delay(100);

// Setting Blue filtered photodiodes to be read
digitalWrite(S2,LOW);
digitalWrite(S3,HIGH);

// Reading the output frequency
frequency = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255
frequency = map(frequency, 25,70,255,0);

// Printing the value on the serial monitor
b = frequency;
delay(100);

}

void loop()

{
  x = analogRead(1) / 330.000;
  y = analogRead(2) / 330.000;
  z = analogRead(3) / 330.000;

  do{
```

```

    for(int i=0;i<4;i++)
    {
        data[i]=mySerial.read();
    }
}while(mySerial.read()!==0xff);

tempc = sensors.getTempCByIndex(0);

mySerial.flush();

if(x>1.0)
{
    if(data[0]==0xff)
    {
        int sum;
        sum=(data[0]+data[1]+data[2])&0x00FF;
        if(sum==data[3])
        {
            distance=(data[1]<<8)+data[2];
            if(distance>30)

{
            if((r>=255&g>=255&b>=255)|(r<-1000&g<-1000&b<-1000))
            {
                Serial.print("Colour less - Water");
                Serial.print(" ");
            }

            else{
                Serial.print("This is not water");
                Serial.print(" ");
            }
            //Serial.print(distance/10);

            d = (distance/10)*(331.3+0.606*tempc)/343;
            //Serial.print(" ");

            if(d>=9.0){
                Serial.println("The container is empty");
            }

            else if(d<=3.0){
                Serial.println("The level is: 400ml");
            }

            else if(d<=3.2){
                Serial.println("The level is: 390ml");
            }

            else if(d<=3.7){

```

```
        Serial.println("The level is: 350ml");
    }

    else if(d<=3.9){
        Serial.println("The level is: 300ml");
    }

    else if(d<=4.1){
        Serial.println("The level is: 270ml");
    }

        else if(d<=4.3){
        Serial.println("The level is: 250ml");
    }

        else if(d<=5){
        Serial.println("The level is: 200ml");
    }

        else if(d<=5.5){
        Serial.println("The level is: 150ml");
    }

        else if(d<=6){
        Serial.println("The level is: 100ml");
    }

        else if(d<=7){
        Serial.println("The level is: 50ml");
    }

        else if(d<=9){
        Serial.println("Time to Refill");
    }

    }

    }

    }

    }
    else{

        Serial.println("The Bottle is inclined");

    }

    delay(100);

}
```

CHAPTER 11

REFERENCES

- [1] B. Murray, “Hydration and physical performance,” *Journal of the American College of Nutrition*, vol. 26, no. sup5, pp. 542S–548S, 2007.
- [2] D. J. Casa, P. M. Clarkson, and W. O. Roberts, “American college of sports medicine roundtable on hydration and physical activity: consensus statements,” *Current sports medicine reports*, vol. 4, no. 3, pp. 115–127, 2005.
- [3] M. N. Sawka, S. J. Montain, and W. A. Latzka, “Hydration effects on thermoregulation and performance in the heat,” *Comparative Biochemistry and Physiology Part A: Molecular & Integrative Physiology*, vol. 128, no. 4, pp. 679–690, 2001.
- [4] J. Shannon, E. White, A. L. Shattuck, and J. D. Potter, “Relationship of food groups and water intake to colon cancer risk.” *Cancer Epidemiology and Prevention Biomarkers*, vol. 5, no. 7, pp. 495–502, 1996.
- [5] M. D. Benefer, B. M. Corfe, J. M. Russell, R. Short, and M. E. Barker, “Water intake and post-exercise cognitive performance: an observational study of long-distance walkers and runners,” *European journal of nutrition*, vol. 52, no. 2, pp. 617–624, 2013.
- [6] L. E. Armstrong, M. S. Ganio, D. J. Casa, E. C. Lee, B. P. McDermott, J. F. Klau, L. Jimenez, L. Le Bellego, E. Chevillotte, and H. R. Lieberman, “Mild dehydration affects mood in healthy young women,” *The Journal of nutrition*, vol. 142, no. 2, pp. 382–388, 2012.
- [7] M. S. Ganio, L. E. Armstrong, D. J. Casa, B. P. McDermott, E. C. Lee, L. M. Yamamoto, S. Marzano, R. M. Lopez, L. Jimenez, L. Le Bellego et al., “Mild dehydration impairs cognitive performance and mood of men,” *British Journal of Nutrition*, vol. 106, no. 10, pp. 1535–1543, 2011.
- [8] C. J. Edmonds, R. Crombie, H. Ballieux, M. R. Gardner, and L. Dawkins, “Water consumption, not expectancies about water consumption, affects cognitive performance in adults,” *Appetite*, vol. 60, pp. 148–153, 2013.
- [9] T. Burkholder, C. Foltz, E. Karlsson, C. G. Linton, and J. M. Smith, “Health evaluation of experimental laboratory mice,” *Current protocols in mouse biology*, pp. 145–165, 2012.
- [10] M. Sawka, “Dietary reference intakes for water, potassium, sodium, chloride, and sulfate. chapter 4-water,” *DTIC Document*, Tech. Rep., 2005.
- [11] T. Hamatani, M. Elhamshary, A. Uchiyama, and T. Higashino, “Fluid meter: Gauging the human daily fluid intake using smartwatches,” *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, p. 113, 2018.

- [12] P. Watson, A. Whale, S. A. Mears, L. A. Reyner, and R. J. Maughan, "Mild hypohydration increases the frequency of driver errors during a prolonged, monotonous driving task," *Physiology & behavior*, vol. 147, pp. 313–318, 2015.
- [13] W. L. Kenney and P. Chiu, "Influence of age on thirst and fluid intake." *Medicine and science in sports and exercise*, vol. 33, no. 9, pp. 1524– 1532, 2001.
- [14] J. Ericson, "'75% of americans may suffer from chronic dehydration, according to doctors." *medical daily*," 2013.
- [15] W. Juan and P. P. Basiotis, "More than one in three older americans may not drink enough water," *Family Economics and Nutrition Review*, vol. 16, no. 1, p. 49, 2004.
- [16] D. Benton and N. Burgess, "The effect of the consumption of water on the memory and attention of children," *Appetite*, vol. 53, no. 1, pp. 143–146, 2009.
- [17] cancer.net, "Dehydration," Mar 2019. [Online]. Available: <https://www.cancer.net/coping-with-cancer/physical-emotional-and-social-effects-cancer/managing-physical-side-effects/dehydration>
- [18] HydraCoach Intelligent Water Bottle, 2011. [Online]. Available: "<https://hydracoach.com/>"
- [19] HidrateSpark Smart Bottles, 2012. [Online]. Available: "<https://hidratespark.com/>"
- [20] "H2opal smart water bottle hydration tracker," 2016. [Online]. Available: <https://www.h2opal.com/>
- [21] Y. Mengistu, M. Pham, H. M. Do, and W. Sheng, "Autohydrate: A wearable hydration monitoring system," in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016, pp. 1857–1862.
- [22] N. Alshurafa, H. Kalantarian, M. Pourhomayoun, J. J. Liu, S. Sarin, B. Shahbazi, and M. Sarrafzadeh, "Recognition of nutrition intake using time-frequency decomposition in a wearable necklace using a piezoelectric sensor," *IEEE sensors journal*, vol. 15, no. 7, pp. 3909– 3916, 2015.
- [23] M. Farooq and E. Sazonov, "Accelerometer-based detection of food intake in free-living individuals," *IEEE sensors journal*, vol. 18, no. 9, pp. 3752–3758, 2018.
- [24] G. Ascioğlu and Y. Senol, "Design of a wearable wireless multi-sensor monitoring system and application for activity recognition using deep learning," *IEEE Access*, vol. 8, pp. 169 183–169 195, 2020.
- [25] H. Griffith and S. Biswas, "Improving water consumption estimates from a bottle-attachable sensor using heuristic fusion," in 2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks"(WoWMoM). IEEE, 2019, pp. 1–3