Santosh Aryal

Smart System Using Particle Photon and Sensors

Metropolia University of Applied Sciences

Bachelor of Engineering

Electronics

Bachelor's Thesis

13 April 2021

Author Title Number of Pages Date	Santosh Aryal Smart System Using Particle Photon and Sensors 40 pages + 1 appendix 13 April 2021
Degree	Bachelor of Engineering
Degree Programme	Electronics
Instructors	Matti Fischer, Principal Lecturer

The main objective of this thesis was to study IoT devices, using Particle photon and different kinds of sensors. The thesis explains the working principle of sensors and the physics behind them with their applications and how these sensors can help to build different types of IoT devices. A system was built using particle photon and sensors which can be used in different applications.

This project has an IoT device called photon which is used as a microcontroller. The photon board is powered by a Cypress Wi-Fi chip alongside a powerful STM32 ARM Cortex M3 microcontroller (STM32F205RGY6 120 MHz). The Wii-Fi module of this device is a Broadcom BCM43362 Wi-Fi chip with 802.11 b/g/n. It also consists of 1MB flash alongside 128KB RAM. For power supply, it consists of an onboard USB micro B connecter and VIN pin.

Different kinds of sensors are fundamental elements to build IoT devices. Ultrasonic sensor, Humidity and temperature sensor, Buzzer, Camera, and OLED display were used in this project.

Ultrasonic sensor is used to measure the proximity of the object using ultrasound, whereas humidity and temperature sensor measures the humidity and temperature which is displayed in the OLED display. A buzzer is used to notify the changes and with the help of the camera, the live view of the surroundings can be seen. Also using an IFTTT platform, notification is sent to the user's phone if any changes occur. The accumulated data can be analyzed from anywhere using ThingSpeak platform.

As a result, smart was system built in this project. The system can be used in different applications like Parking Assistance, Anti-Theft Alarm, Automatic Water Leveling, and Monitoring System analyze the live data and control the devices wirelessly, and many more.

Keywords	IoT, Sensor, Wireless, Particle Photon, IFTTT, TTL Camera

Contents

List of Abbreviations

List of Figures

1 Introduction 1

2 Particle Photon 2

2.1 Overview 2
2.2 Interface 3
2.2.1 Microcontroller 3
2.2.2 RF 3
2.2.3 Peripherals and GPIO 4
2.2.4 Status Led 6
2.3 Software 6
2.3.1 Developer Tools 6
2.4 Communication 10
2.4.1 I2C Communication 10
2.4.2 SPI Communication 11

3 If This, Then That 13

4 Sensors and Devices Used 14

4.1 LCD Display 14
4.2 Ultrasonic Sensor 15
4.2.1 Overview 15
4.2.2 Ultrasonic Sensor HC-SR04 16
4.2.3 Testing of HC-SR04 18
4.3 Humidity and Temperature Sensor (DHT22) 23
4.3.1 Capacitive Humidity Sensor 23
4.3.2 Negative Temperature Coefficient Thermistor (NTC) 26
4.4 OLED 27
4.5 Piezo Buzzer 28
4.6 TTL Serial Camera 29

5 System Design 29

5.1 Creating Applets and Webhook for Data Analysis 32

5.2 TTL Camera Setup 35

5.2.1 Running on Local Machine 35

6 Conclusion 38

References 40

Appendices

Appendix 1.

Code for Humidity and Temperature Sensor, OLED display, and Ultrasonic Sensor

List of Abbreviations

BAW	Bulk Acoustic Wave
CAN	Controller Area Network
CLI	Command Line Interface
DAC	Digital to Analog
FCC	Federal Communication commission
GPIO	General Purpose Input/output
IoT	Internet of things
IFTTT	If this then that
LCD	Liquid Crystal Display
MISO	Master In Slave Out
MOSI	Master Out Slave In
NTC	Negative Temperature Coefficient
PCB	Printed Circuit Board
PWM	Pulse Width Modulation
RAM	Random Access Memory
RF	Radio Frequency
RTC	Real-time Clock
TTL	Transistor–transistor logic
USB	Universal Service Bus
VBAT	Battery Voltage
Wi-Fi	Wireless Fidelity

List of Figures

Figure 1: Overview of particle photon. Reprinted from (1) 2
Figure 2: Peripherals and GPIO. Reprinted from (2) 4
Figure 3: PØ board with Pin marking. Reprinted from (2) 5
Figure 4: Web IDE Particle Photon. Reprinted from (3) 7
Figure 5: Tinker Interface. Reprinted from (5) 9
Figure 6: Connection diagram. Reprinted from (6) 10
Figure 7: Writing to Slave Register. Reprinted from (7) 11
Figure 8: Reading from the slave. Reprinted from (7) 11
Figure 9: Single master multiple salves configuration. Reprinted from (8) 12
Figure 10: SPI 4-Pins Interface. Reprinted from (9) 13
Figure 11: Grove LCD Display. Reprinted from (10) 14
Figure 12: Principle of active sonar. Reprinted from (11) 15
Figure 13: Ultrasonic Sensor HC-SR04 (12) 16
Figure 14: Pulse diagram of HC-SR04 17
Figure 15: Hardware Implementation 18
Figure 16: Particle Console Showing Distance in Real-Time 21
Figure 17: Echo pulse observed in Oscilloscope 22
Figure 18: Grove- Humidity and Temperature Sensor (13) 23
Figure 19: Structure of Capacitive Sensor. Reprinted from (14) 24
Figure 20: Inside a Capacitor. Reprinted from (15) 25
Figure 21: Temperature Vs Resistance (Left) NTC thermistor (Right) (18) 26
Figure 22: OLED Display Reprinted from (19) 27
Figure 23: Piezo Buzzer. Reprinted from (20) 28
Figure 24: TTL Camera Module. Reprinted form (22) 29
Figure 25: System Layout 30
Figure 26: System Design 31
Figure 27: IFTTT Applets 32
Figure 28: Phone notification 33
Figure 29: Webhook created using Particle IDE 34
Figure 30: Thingspeak Channel 34
Figure 31: Getting Particle Token Using VS Code CLI 35
Figure 32: IP SERVER and Port 35
Figure 33: Folder that contains app.js 36
Figure 34: Changing access token 36

Figure 35: TTL Camera Interface 37
Figure 36: Picture taken from TTL Camera 38

List of Tables

Table 1: Measured Distance 21

Listings

Listing 1: Code for Ultrasonic sensor Application 20

1 Introduction

With the advancement of technology, the world has been changing and you can control everything with a single button in your hand. In terms of technology, for the past four decades, there has been an explosion of ideas and innovations. Over the years, many ideas about smart devices have been emerged and have been developed by many companies to make our day-to-day life easier. Wouldn't it be easier to control the whole house with the use of a mobile phone which is always nearby you? Multiple technologies like machine learning, real-time analytics, sensors, and embedded systems have coalesced and smarts devices were developed. Later, to make it easier the term IoT (Internet of things) was likely coined by Kevin Ashton and later MIT's Auto-ID Center in 1999. Internet of things (IoT) is one of the hot topics being discussed at the moment because of its wide range of applications in the consumer, industrial and commercial field.

IoT devices are part of the greater concept of Home Automation, which allows controlling the heating, lighting, media, security systems, and air conditioning, automobile, embedded systems, and many more. IoT is a huge subject and touches many fields of technology.

The goal of the project was to build a smart system that can be used for different applications like parking assistance, home security, and analyzing and monitoring system for the water reservoir facility.

For this project, as the main system, a Particle Wi-Fi module as a micro-controller was used. It provides a very powerful device with wireless data transmission and it is powered by STM32 (120 MHz) ARM Cortex M3. For wireless data transmission over the network, it uses Broadcom BCM43362 Wi-Fi chip (Cypress Wi-Fi Chip) which provides the highest level of integrating for mobile and handheld devices.

2 Particle Photon

2.1 Overview

The particle combines a powerful ARM Cortex M3 microcontroller with a Broadcom Wi-Fi chip in a tiny thumbnail-sized module called the PØ (P-zero). The particle has a solid 3.3 VDC switch mode power supply (SMPS), RF, and user interface components to the board on a small single-sided printed circuit board called the Photon.

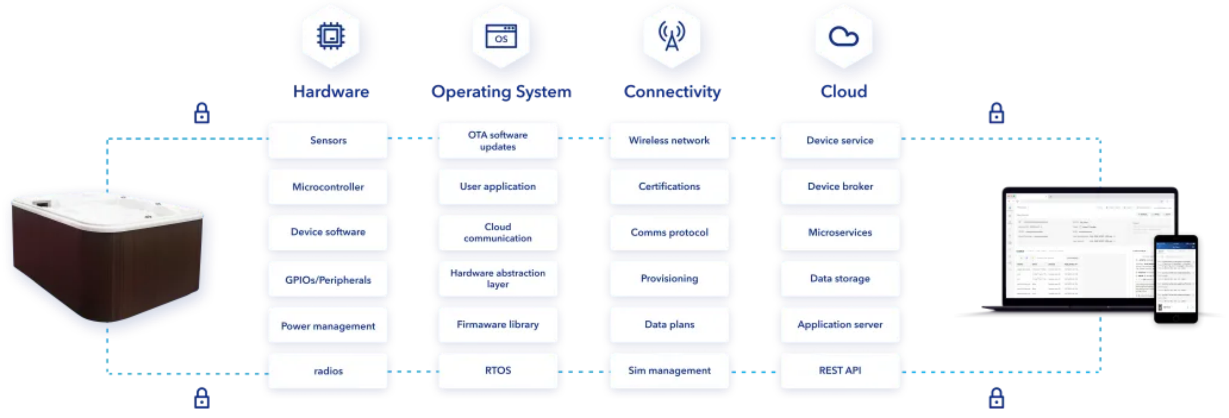


Figure 1: Overview of particle photon. Reprinted from (1)

As shown in figure 1, particle photon provides a wide range of hardware to work with. It has a robust operating system that is mostly used in embedded IoT devices and provides an easy way to create logic and control the devices. Particle photon also provides connectivity with reliable cloud infrastructure to monitor the data easily with hardware and cloud security, keeping user privacy intact.

2.2 Interface

2.2.1 Microcontroller

A microcontroller (µC or MCU) is a computer that controls other parts of an electronic system. It is a single chip of metal-oxide-semiconductor integrated circuit. A microcontroller comprises a CPU with possibilities of multiple CPUs alongside a memory unit and different input/out peripherals which can be programmed. Microcontrollers are used in many automated devices and products, such as remote controls, embedded systems, robots, mobile devices, and almost every electronic device has a microcontroller.

In particle photon, STM32 ARM Cortex M3 is used as a microcontroller that acts as the brain of the photon board along with Broadcom Wi-Fi chip for wireless connection. The microcontroller used in the photon board is from the STM32F20 family which is based on a high-performance Cortex M3 operating core with a frequency of 120 MHz and supports up to 15 communication interfaces.

The Photon board is powered with an onboard USB Micro B connector and VIN pin. However, powering the board directly from the VIN pin needs voltage regulated between 3.6 VDC and 5.5 VDC. When it is powered via a USB port VIN will output a voltage of 4.8 VDC. The average current consumption is 80 mA with 5 V at VIN with Wi-Fi on. In the particle photon board, input wires are kept as short as possible to avoid voltage spikes.

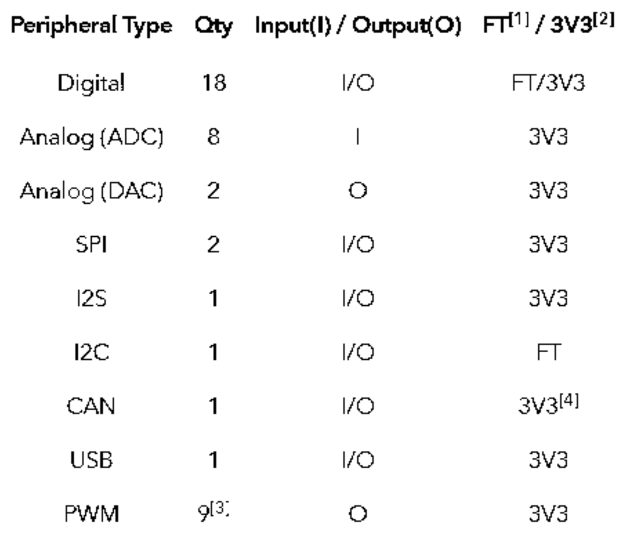
2.2.2 RF

RF modules accomplish communications through their signal processing, Wi-Fi, ZigBee, Bluetooth, Radio transceiver, Duplexer, and BAW. In Photon, the radio frequency module is a finely tuned impedance-controlled network of components that optimize the efficacy and sensitivity of wireless communications. PØ module consists of three RF ports that have a 10 pF RF quality DC blocking capacitor that effectively passes 2.4 GHz frequencies while blocking unwanted voltages which may damage the RF of the device.

The board consists of two FCC (Federal Communication Commission) approved antennas. Dipole antenna manufactured by Lumen Radio has a gain of 2.15 dBi and there is a Chip antenna manufactured by Advance Ceramic X that has a gain of 1.3 dBi.

2.2.3 Peripherals and GPIO

Photon consists of different options with analog, digital, and communication interfaces. To connect sensors, buttons, lights, buzzers, motors the GPIO pins can be utilized.



Peripheral Type	Qty	Input(I) / Output(O)	FT[1] / 3V3[2]
Digital	18	I/O	FT/3V3
Analog (ADC)	8	I	3V3
Analog (DAC)	2	O	3V3
SPI	2	I/O	3V3
I2S	1	I/O	3V3
I2C	1	I/O	FT
CAN	1	I/O	3V3[4]
USB	1	I/O	3V3
PWM	9[3	O	3V3

Figure 2: Peripherals and GPIO. Reprinted from (2)

As shown in figure 2, photon board has different possibilities for communication. It consists of two SPI interfaces, one I2S interface, and one I2C interface. It also consists of a CAN (Controller Area Network) bus and PWM (Pulse-width modulation) pin which makes it versatile for different applications in the IoT sector.

The photon board consists of 24 pins on the outside with extra 7 pads on the bottom which can be used to connect to extra signals if necessary.

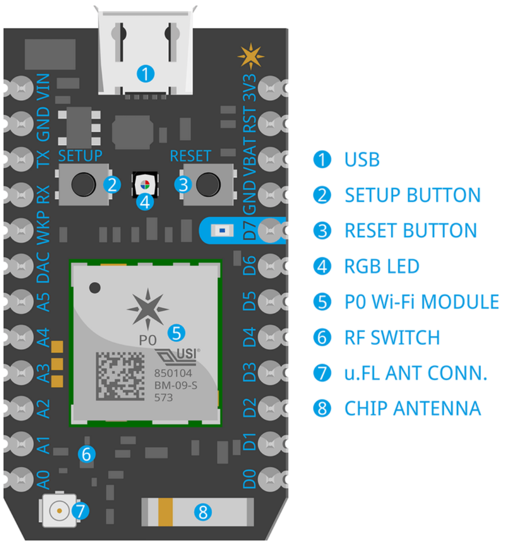


Figure 3: PØ board with Pin marking. Reprinted from (2)

Figure 3 shows all the pin marking available in the photon board. VIN pin at the top left can be used as either input or output. When the pin is used as input, it can supply the power of 3.6 to 5.5 VDC to the photon board. As an output pin, it delivers the voltage of 4.8 VDC (photon should be powered by USB) with a max load of 1 A in VIN. 3V3 pin is the regulated 3.3 V supply.

RST is the pin for active-low reset input. VBAT (Battery Voltage) is an input-only hardware-based power mode that maintains critical operations when a power loss occurs on 3.3 VDC. When it is connected to a backup supply, it allows the RTC (Real-time Clock) and some SRAM to continue operating when the main supply is removed. DAC is a 12-bit Digital to Analog output pin.

WKP is an active-high wake-up pin. When the modules are on sleep or standby modes, the pin is used to wake up the modules. It can also be used as a digital GPIO or ADC. A0-A7 are 12-bit Analog to Digital inputs, also digital GPIOs. D0-D7 are digital-only GPIO-pins. Whereas A4, A5, A7, D0, D1, D2, D3 can also be used as PWM output.

2.2.4 Status Led

The Photon board has a status led as shown in figure 3 (4 RGB LED) which tells in which mode the board is in its current state. The device can be put in a different state using two buttons mounted on the board SETUP and RESET as shown in figure 3. Status LED has different types of blinking light for different modes. When the board is connected to the internet it will be breathing cyan and if the board is trying to connect to Wi-Fi it will be blinking green. Blinking blue means the board is in a listening mode which is done by holding the SETUP button for 3 seconds but if the LED is breathing blue state, it means the device is not connected to Wi-Fi. When the firmware of the board crashes it will blink in a pattern of more than 10 red blinks.

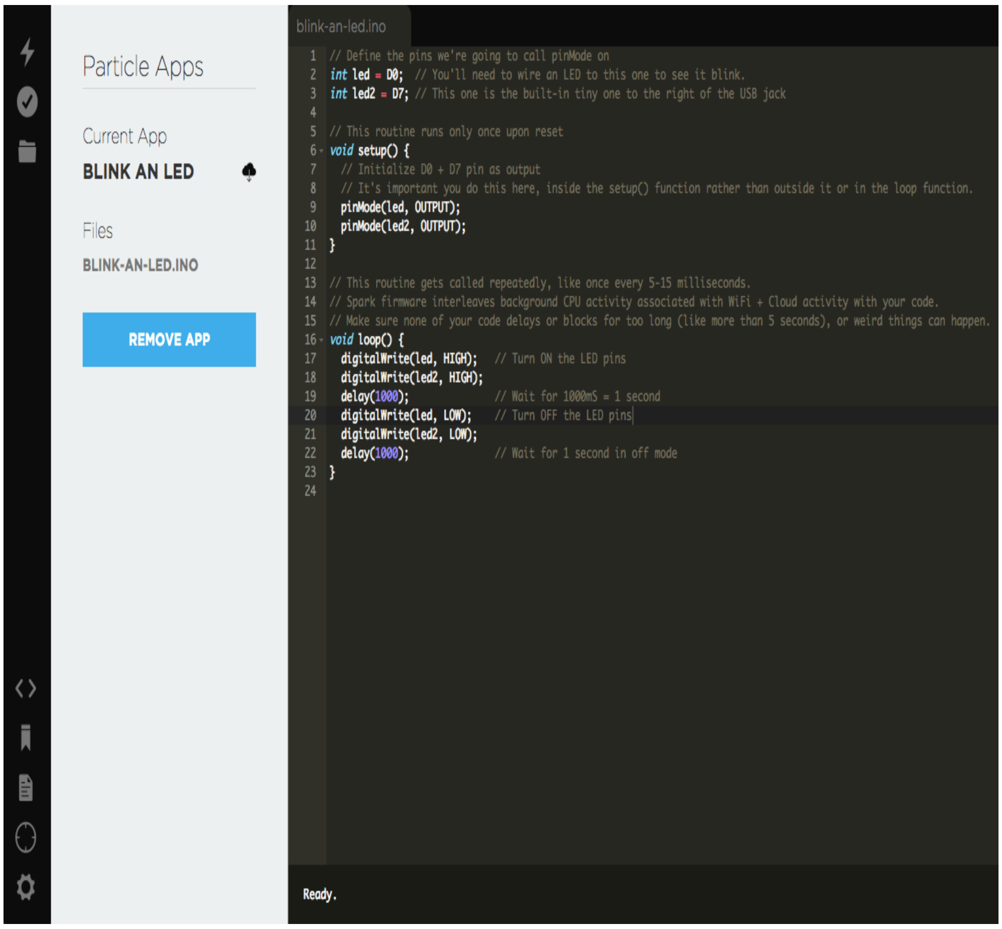
2.3 Software

2.3.1 Developer Tools

Alongside the hardware, particle photon provides the software to obtain the desired results. Particle photon provides different platforms for executing and debugging the code for the hardware. From web app to console, particle photon gives the developer edge to start the different projects.

2.3.1.1 Particle Web IDE

Particle Web IDE provides the ability to program the particle devices by providing the platform to write code in the browser, without installation of any development tool. Particle provides a different platform to program the device, but Web IDE is the easiest way to perform the task. It has a user-friendly interface with different options to make the job easier and less time-consuming. To be able to use the Web IDE one must have a particle account and need to login into the account which means there must be an internet connection to use this feature (3).



```
// Define the pins we're going to call pinMode on
int led = D0;  // You'll need to wire an LED to this one to see it blink.
int led2 = D7; // This one is the built-in tiny one to the right of the USB jack

// This routine runs only once upon reset
void setup() {
  // Initialize D0 + D7 pin as output
  // It's important you do this here, inside the setup() function rather than outside it or in the loop function.
  pinMode(led, OUTPUT);
  pinMode(led2, OUTPUT);
}

// This routine gets called repeatedly, like once every 5-15 milliseconds.
// Spark firmware interleaves background CPU activity associated with WiFi + Cloud activity with your code.
// Make sure none of your code delays or blocks for too long (like more than 5 seconds), or weird things can happen.
void loop() {
  digitalWrite(led, HIGH);   // Turn ON the LED pins
  digitalWrite(led2, HIGH);
  delay(1000);               // Wait for 1000mS = 1 second
  digitalWrite(led, LOW);    // Turn OFF the LED pins
  digitalWrite(led2, LOW);
  delay(1000);               // Wait for 1 second in off mode
}
```

Figure 4: Web IDE Particle Photon. Reprinted from (3)

Figure 4 shows the user interface of the Particle Web IDE. In figure 4, the three buttons on the top left corner are the most used button with important functions. The first one is Flash, which runs the current code and flashes it to the particle device. Verify is the second button that helps the user to compile the code without flashing it to the device. This helps the user to find the error in the code, which is shown in the debug console at the bottom of the screen. And the third button helps to save any changes made to the code. There are four more buttons on the bottom left corner which are used for the basic navigation through the IDE and browse through documentation and library.

2.3.1.2 CLI Command Interface

Another way to program the photon board is using the command-line interface (CLI), which is a powerful tool to communicate with the device and the particle cloud. Using

CLI, everything can be done from creating an account or logging in or setting up Wi-Fi to flashing code and debugging. To install the CLI in macOS and Linux the easiest way is to open the terminal and type **bash <(curl –sL https://partilce.io/install-cli** which installs the Node.js module that contains the CLI code. For Windows users, Windows CLI Installer can be downloaded and run to install the Particle CLI (4).

A function can be called to do the specific task in CLI. For example, to go the setup mode **particle setup** can be called and similarly, **particle login** can be called to login into the particle account. CLI also supports using libraries for the project which allows the integration of already written code in the project and helps speed up the project. To find the library **particle library search** can be called and to add the library **particle library add** can be called and to compile the code **particle compile** can be called.

2.3.1.3 Mobile Application

Particle photon provides the mobile application **Particle IoT** for the users, which makes learning easier without the need to write the code, useful for the starters. Particle IoT applications can be used to claim the photon device and connect it to the internet. The particle mobile application can be downloaded without any cost for any device. After downloading the application and connecting the photon to the internet, the tinker section inside the Particle Mobile Application can be used (5).

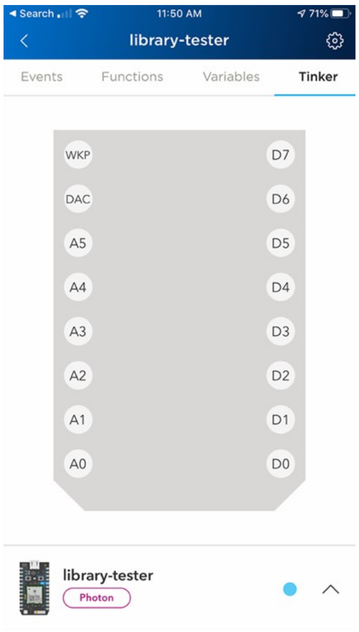


Figure 5: Tinker Interface. Reprinted from (5)

As shown in figure 5 this application provides an easy interface to the user with the four possible functions. Each pin can write and read the analog and digital signals. The four possible functions provided are digitalWrite, analogWrite, digitalRead, and analogRead. The easiest way to use the pin is just to tap on the screen and it will show the options to choose from.

2.3.1.4 Particle Console

The particle provides a platform to manage and interact with the particle's device easily. It has a user-friendly interface with various features. Using particle console, specific data of the particle devices can be seen including unique Device ID, last active time with last known location of the device. It has an event log section where real-time information from the particle devices can be analyzed with the option of data filtering. The most important feature of the particle console is integrations which allow a user to send data to external tools and services from the particle devices. Integration can be used with different IoT platforms like Azure IoT Hub, Google Cloud Platform, and Webhook provided by Math-Works.

2.4 Communication

2.4.1 I2C Communication

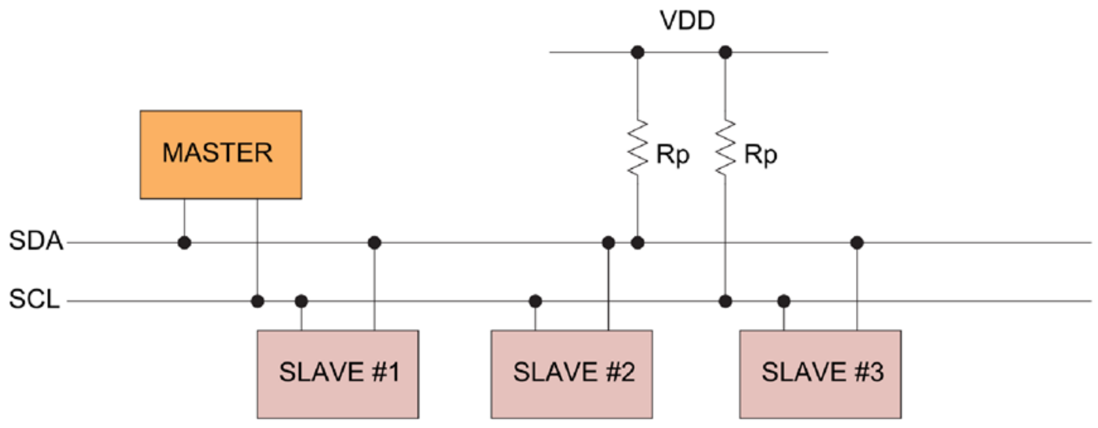


Figure 6: Connection diagram. Reprinted from (6)

I2C (Inter-Integrated Circuit) is a serial half-duplex protocol for a two-wire interface to connect low-speed devices like microcontrollers, EEPROMs, A/D, and D/A converters, I/O interfaces, and other similar peripherals in embedded systems. It was invented by Philips and now it is used by almost all major IC manufacturers (6).

The I2C bus is very famous because it is powerful and simple to use with two wires: SCL (serial clock) and SDA (serial data) with pull-up resistors as shown in Figure 6. In I2C hardware, the master is the device that starts the communication and drives the clock (SCL) line whereas the slave responds to the master and acts accordingly. In the I2C protocol, there can be more than one master to connect and control the multiple numbers devices. In an electronics device, a microprocessor acts as a master whereas LCDs, EEPROM acts as a slave. In this type of communication, the data frame is 8 bits long and starts with the most significant bit which is flowed by acknowledge bit/not acknowledge bit (ACK/NACK) which verifies whether the data has been received successfully. In particle photon, D1 is SCL pin and D0 is SDA pin which is used for I2C communication.

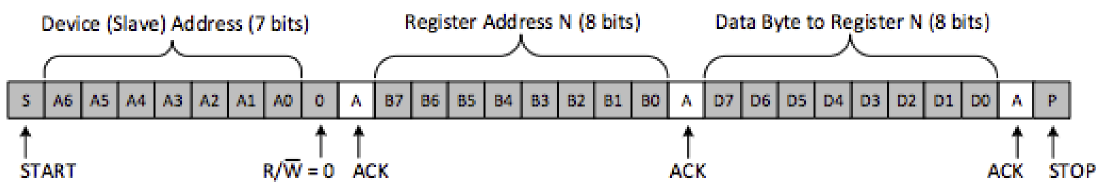

Device (Slave) Address (7 bits)
Register Address N (8 bits)
Data Byte to Register N (8 bits)
S
A6
A5
A4
A3
A2
A1
A0
0
A
B7
B6
B5
B4
B3
B2
B1
B0
A
D7
D6
D5
D4
D3
D2
D1
D0
A
P
START
R/W̄ = 0
ACK
ACK
ACK
STOP

Figure 7: Writing to Slave Register. Reprinted from (7)

Figure 7 shows how the master writes to the slave. The communication starts with the condition followed by the device (slave) address to which the data to be written. After the device address, the last bit is set to 0 which means write followed by acknowledge bit. If the device address is correct, the master sends the register address to which the master wishes to write. After the registered address, the master sends all the data to be written until the STOP condition occurs which terminates the communication. Reading from slaves also works similarly with some extra steps (7).

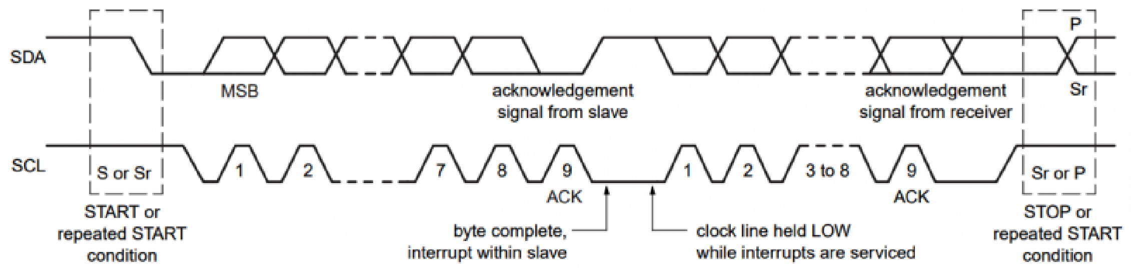

SDA
SCL
MSB
acknowledgement signal from slave
acknowledgement signal from receiver
P
Sr
S or Sr
1
2
7
8
9
ACK
1
2
3 to 8
9
ACK
Sr or P
START or repeated START condition
byte complete, interrupt within slave
clock line held LOW while interrupts are serviced
STOP or repeated START condition

Figure 8: Reading from the slave. Reprinted from (7)

In figure 8, clock and data lines are shown where the communication starts when the data line is low while the clock line is still high.

2.4.2 SPI Communication

SPI stands for Serial Peripheral Interface which is also a full-duplex synchronous serial communication interface used for a short distance, generally to established communication between microcontrollers and sensors, ADCs, DACs, registers, and other types of peripheral devices (8). In SPI, the clock signal (SCK) is generated only by the master. In this type of communication, there can be only one master but there can be multiple slaves. SPI is a four-wire communication system that allows transferring 2 to 16 bits of data into and out of the device.

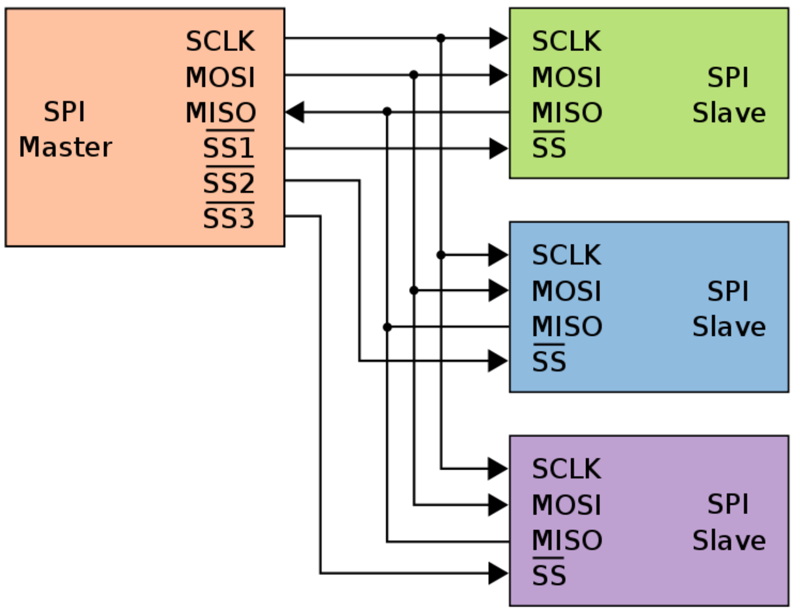


Figure 9: Single master multiple salves configuration. Reprinted from (8)

As shown in figure 9 it has four main bus wires Master-Out, Slave-In (MOSI), Master-In, Slave-Out (MISO), System Clock (SCLK), and Slave Select (SS). MOSI carries the data out of the master and transfers it to the slave whereas MISO does the opposite by carrying data out of slave to master. Sometimes MOSI can be written as SIMO and MISO can be written as SOMI and slave select can be written as SPI_CS but they mean the same thing. Since SPI can have multiple slaves so it has a select slave option to choose the slave to which communication needs to be built.

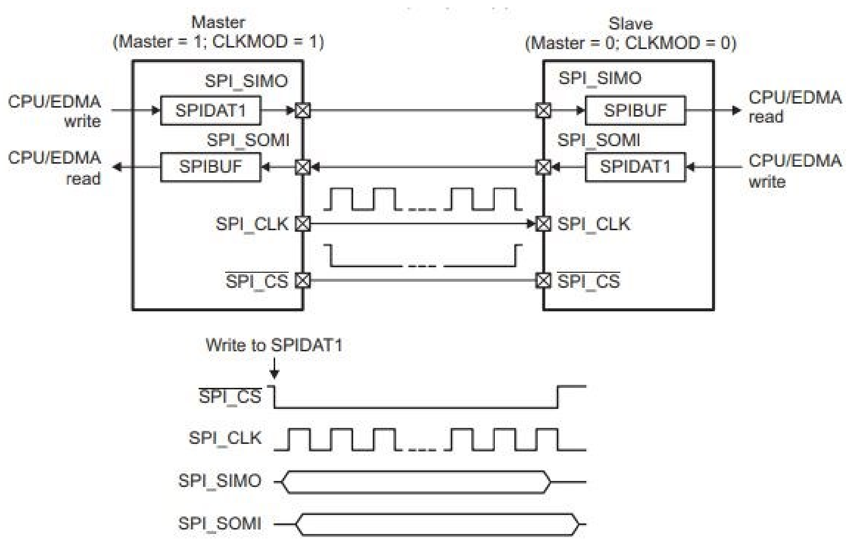


Figure 10: SPI 4-Pins Interface. Reprinted from (9)

In SPI communication, the data transmission begins with the master sending the clock signal (SPI_CLK to slave and master switches select slave to low which triggers the slave as shown in figure 10. When the slave is active, the master starts sending data one bit at a time using the MOSI line. The data is received in the same way as they were sent by the master and if the slave wants to communicate with the master, the MISO line is used. In particle photon, there are two sets of SPI pins A5 to A2 and D5 to D2. Pins A5 and D2 are MOSI, A4 and D3 are MISO, A3 and D4 are SCK, and A2 and D5 are SS pins.

3 If This, Then That

IFTTT (if this, then that) is a web-based service provider that allows users to create a loop of conditional situations triggered by changes which can be sensors or within web services such as Gmail, Instagram, Facebook, etc. An applet, for example, sends an email message to the user if the IR sensor of the user's house detects motion. IFTTT is used to automate web application tasks, IoT applications, home automation, and many more. It is easy to use with a mobile application or directly from the web page after creating a free account. Big companies like BMW, Facebook, Domino, and Ring are using IFTTT to create applets on the IFTTT platform.

In this project, a conditional statement (applet) was created to send the notification to the user’s phone whenever the ultrasonic sensor picks up any changes.

4 Sensors and Devices Used

4.1 LCD Display

For testing ultrasonic, a display, a new version of Grove- 16 x 2 LCD is used which is easy to handle with 4 pins and I2C communication. It has 16 columns with 32 characters in total. As shown in figure 11, it has 2 signals pins SDA and SCL alongside 2 power pins VCC and ground. It has a super-twisted nematic display with an onboard microcontroller that operates in the voltage range of 3.3 V to 5 V. I2C communication makes it compatible with many boards like Arduino, Raspberry Pi, and with Particle Photon. The new version of the grove display already has I2C pull-up resistors between VCC and SCL and pins and between VCC and SDA pins whereas in older versions the resistors were missing (10).

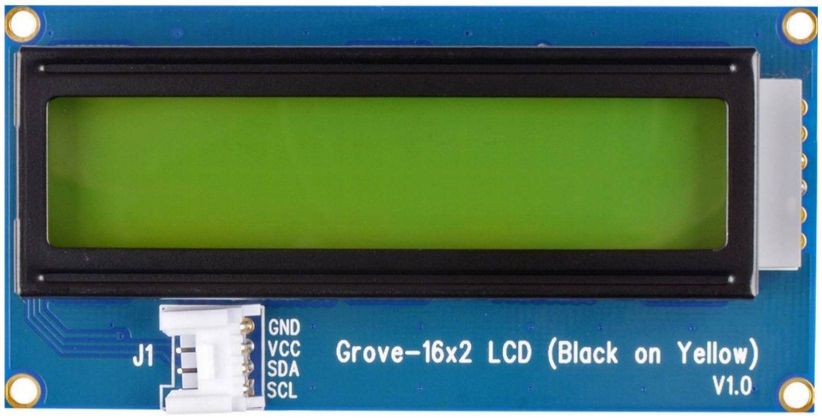


Figure 11: Grove LCD Display. Reprinted from (10)

4.2 Ultrasonic Sensor

4.2.1 Overview

Sound is a longitudinal wave that can be characterized in terms of frequency, wavelength, velocity, amplitude, or speed of propagation. The strength of the sound wave is measured in decibels (dB). The velocity of sound depends upon many environmental factors like temperature, pressure, elasticity, and density of the medium through which it is traveling. Sound wave also shows the phenomenon of reflection, refraction, diffraction, and interference.

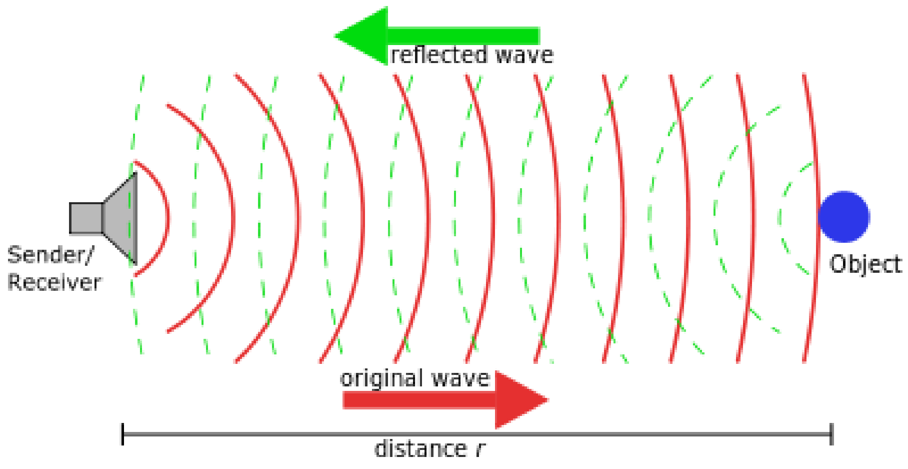


Figure 12: Principle of active sonar. Reprinted from (11)

Ultrasound is sound waves with frequencies higher than the upper audible limit 20 kHz range of the human hearing (11). When such sound waves are fabricated and sent to a certain direction as shown in Figure 12, they tend to reflect and those reflected sound waves can be used in many different devices to detect the proximity of the object. This technology is like how bats use echolocation to move around without crashing into an obstacle. When all things are put together ultrasonic sensor can be developed. The ultrasonic sensor has many applications in the different industrial sectors. It can be used in parking assist, ADAS (Advance Driver-Assistant Systems), robotics, and other applications where proximity and position sensing is important.

4.2.2 Ultrasonic Sensor HC-SR04

The most common and cheap ultrasonic sensor available in the market is HC-SR04 which uses the same principle as SONAR and RADAR. It can provide 2 cm to 400 cm of range with a measuring angle of 15 degrees to give measurement with high accuracy. HC-SR04 is easy to use with a working voltage of 5 volts DC and a current of 15 mA.

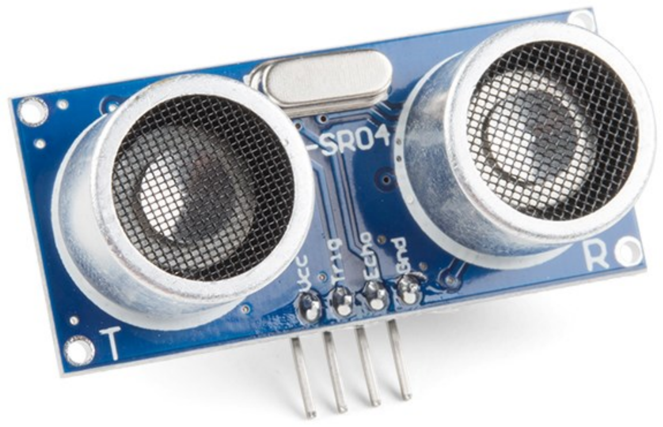


Figure 13: Ultrasonic Sensor HC-SR04 (12)

As shown in figure 13, it consists of four pins, VCC, Trigger, Echo, and Ground. HC-SR04 has two ultrasonic transducers which act as transmitter and receiver. It also has a crystal oscillator at the top which is used to produce the signal.

VCC is for power supply which is 5 volts and a GND pin was used to ground the circuit. A trigger pin was used to produce a 10 μs pulse which triggers the ultrasonic sound pulse.

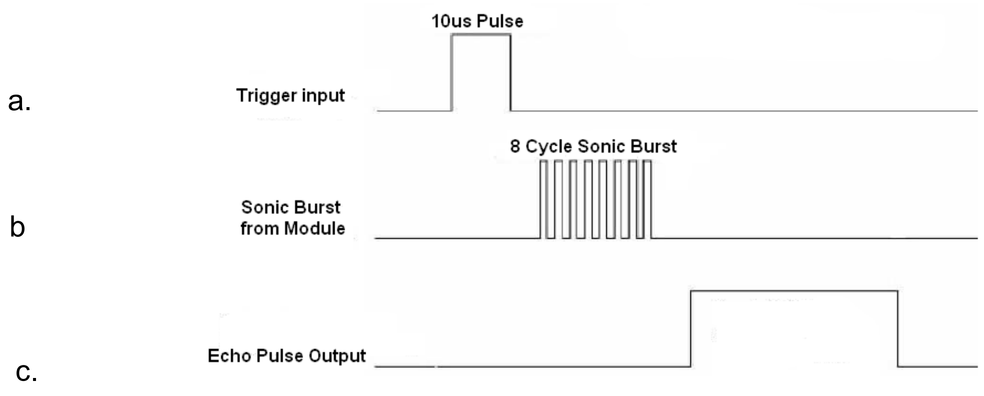


Figure 14: Pulse diagram of HC-SR04

Figure 14 a shows, 10 µs pulse was applied to the trigger pin the sensor transmits the sonic burst of eight pulses at 40 kHz using a transmitter and echo pins go HIGH. When the 8 pulses of ultrasonic sound reflect after striking the surface are received by the receiver, the echo pin goes LOW. The time when the echo pin was HIGH was used to calculate the distance of the object as shown in figure 14 c.

For example, suppose there was an obstacle at an unknown distance and the echo pulse stayed HIGH for 600 µs. Now to calculate the distance following equation can be used.

$$\text{Distance} = \text{Velocity} * \text{Time} \tag{1}$$

The velocity of sound in air is 344 m/s which is 0.0344 cm/µs and the time by the pulse to travel back and forth is 600 µs. The pulse sent out by the transmitter reached the obstacle and reflected so the total time taken by the pulse needs to be divided by 2 to get the correct distance. Using equation 1 and using the values we get.

$$\text{Distance} = (0.0344\ \text{cm}/\ \mu\text{s} * 600\ \mu\text{s})\ /\ 2 \tag{2}$$

$$\text{Distance} = 10.32\ \text{cm} \tag{3}$$

So, the obstacle was at 10.32 cm.

4.2.3 Testing of HC-SR04

Hardware implementation of the Particle Photon board and HC-SR04 with Grove LCD is easy and simple. The connection between these three components is illustrated in figure 15. The following circuit was designed as a test circuit for the HC-SR04 sensor.

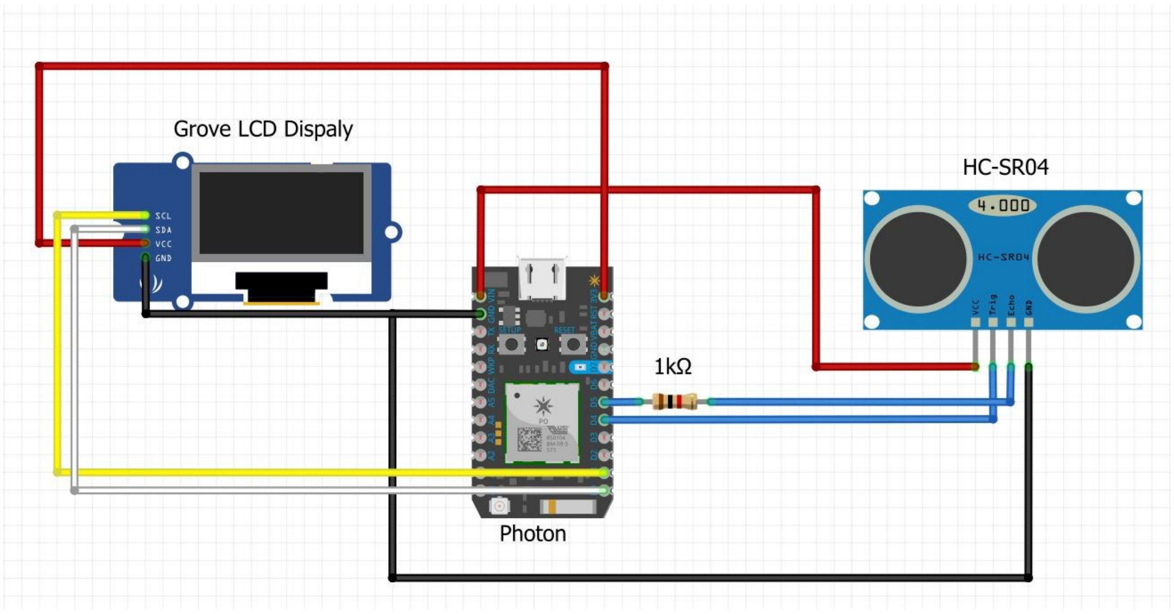


Figure 15: Hardware Implementation

For the HC-SR04, the VCC pin is connected to the Vin pin of Photon, Trigger pin is connected to D4 of the photon, Echo pin connected to D5 with 1 kΩ resistor in series and GND pin is connected GND pin of the photon which completes the connection for the sensor.

In Grove LCD, VCC and GND pins were connected to VCC and GND pin of photon respectively. In Grove display, I2C protocol was used for communication with microcontroller where D1 pin was used as SCL and D0 pin was used as SDA. So, D0 and D1 pins of photon were connected to the SDA and SCL pin of the display respectively.

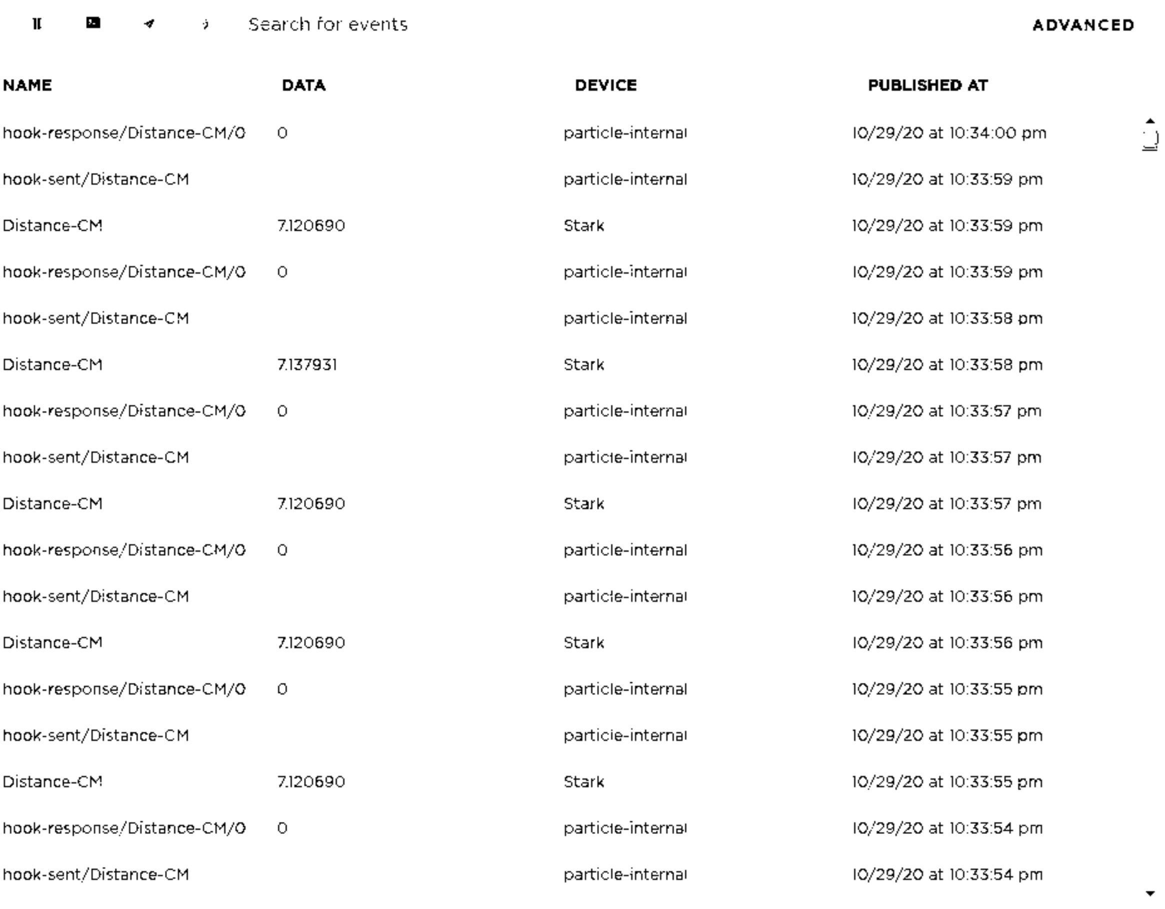
As particle photon provides the Web IDE to program the device for the desired output. In Web IDE, all the information about the photon device can be found. To create a new application, it can be done by clicking the “Create New App” button and give the description of the program, and press enter.

The code written for the configuration shown in figure 15 is given below in Listing 1.

```
#include<HC-SR04.h>
#if defined (PARTICLE)
#else
#include <Wire.h>
#endif

#include "Grove_LCD_RGB_Backlight.h" // including the library for display

rgb_lcd lcd;

const int colorR = 255; // setting the color of the display
const int colorG = 240;
const int colorB = 230;
int trigPin = D4;
int echoPin = D5;

int maxDist = 200;
int minDist = 0;

double distCm = 0.0;


double getDistanceCM() // function to calculate the distance
{
    sendTriggerPulse(trigPin);
    waitForEcho(echoPin, HIGH, 100);
    long startTime = micros();
    waitForEcho(echoPin, LOW, 100);
    long endTime = micros();
    long duration = endTime - startTime;
    double distance = duration / 29.0 / 2.0;
    if (distance < minDist || distance > maxDist)
    {
        return -1;
    }
    return distance;
}

void sendTriggerPulse(int pin) //starts the trigger pin
{
    digitalWrite(pin, HIGH);
    delayMicroseconds(10);
    digitalWrite(pin, LOW);
}

void waitForEcho(int pin, int value, long timeout) // waits for pulse
{
    long giveupTime = millis() + timeout;
    while (digitalRead(pin) != value && millis() < giveupTime)
    {
    }
}

void setup()
{
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    lcd.begin(16, 2);

    lcd.setRGB(colorR, colorG, colorB);
    lcd.print("Distance in CM:");

    delay(1000);

}

void loop() // creating a loop to calculate the distance continuously
{
    distCm = getDistanceCM();
    Particle.publish("Distance-CM", String(distCm));

    delay(1000);


    lcd.setCursor(0, 1);

    lcd.print(String(distCm));

    delay(100);
}
```

Listing 1: Code for Ultrasonic sensor Application

As shown in Listing 1, photon libraries were used for I2C communication, sensor, and grove display. In the first section of the code, the LCD was initialized by using a constant integer variable with trigger pins and echo pins. In the middle section of the code a function, `getdDistanceCM(),` was created to calculate the distance in centimeters. Also, a `void sendTriggerPulse (int pin)` function is created to send 10 µs pulse. To read the returning ultrasound function was written for echo pin `void waitForEcho(int pin, int value, long timeout).` In `void setup()` input and output pins were defined with Grove LCD whereas in `void loop(), getDistanceCM()` was called and `lcd.print(String(distCm)` was called which printed the distance in Grove display.



NAME	DATA	DEVICE	PUBLISHED AT
hook-response/Distance-CM/0	0	particle-internal	10/29/20 at 10:34:00 pm
hook-sent/Distance-CM		particle-internal	10/29/20 at 10:33:59 pm
Distance-CM	7.120690	Stark	10/29/20 at 10:33:59 pm
hook-response/Distance-CM/0	0	particle-internal	10/29/20 at 10:33:59 pm
hook-sent/Distance-CM		particle-internal	10/29/20 at 10:33:58 pm
Distance-CM	7.137931	Stark	10/29/20 at 10:33:58 pm
hook-response/Distance-CM/0	0	particle-internal	10/29/20 at 10:33:57 pm
hook-sent/Distance-CM		particle-internal	10/29/20 at 10:33:57 pm
Distance-CM	7.120690	Stark	10/29/20 at 10:33:57 pm
hook-response/Distance-CM/0	0	particle-internal	10/29/20 at 10:33:56 pm
hook-sent/Distance-CM		particle-internal	10/29/20 at 10:33:56 pm
Distance-CM	7.120690	Stark	10/29/20 at 10:33:56 pm
hook-response/Distance-CM/0	0	particle-internal	10/29/20 at 10:33:55 pm
hook-sent/Distance-CM		particle-internal	10/29/20 at 10:33:55 pm
Distance-CM	7.120690	Stark	10/29/20 at 10:33:55 pm
hook-response/Distance-CM/0	0	particle-internal	10/29/20 at 10:33:54 pm
hook-sent/Distance-CM		particle-internal	10/29/20 at 10:33:54 pm

Figure 16: Particle Console Showing Distance in Real-Time

After the assembly of the hardware and flashing the code in particle photon using Web IDE, the distance measured by the sensor can be perceived in the particle console as shown in figure 16.

Table 1: Measured Distance

Actual Distance (cm)	Echo Time(us)	Distance calculated(cm)	Photon Console (cm)	Error(%)
5	312	5.304	5.8	14
10	720	12.24	12.4	19
15	1060	18.02	18.4	33
20	1300	22.1	22.1	10
30	1760	29.92	30.6	2
60	3460	58.82	59.3	2

After some tests, values were acquired as shown in Table 1. In the test, the echo time was recorded alongside the distance shown by the particle console as shown in Table

1. Using the echo time, the distance was calculated, which were quite accurate values compared to console values. While measuring the distance many factors affect the measurement so different values can be obtained for the constant distance.

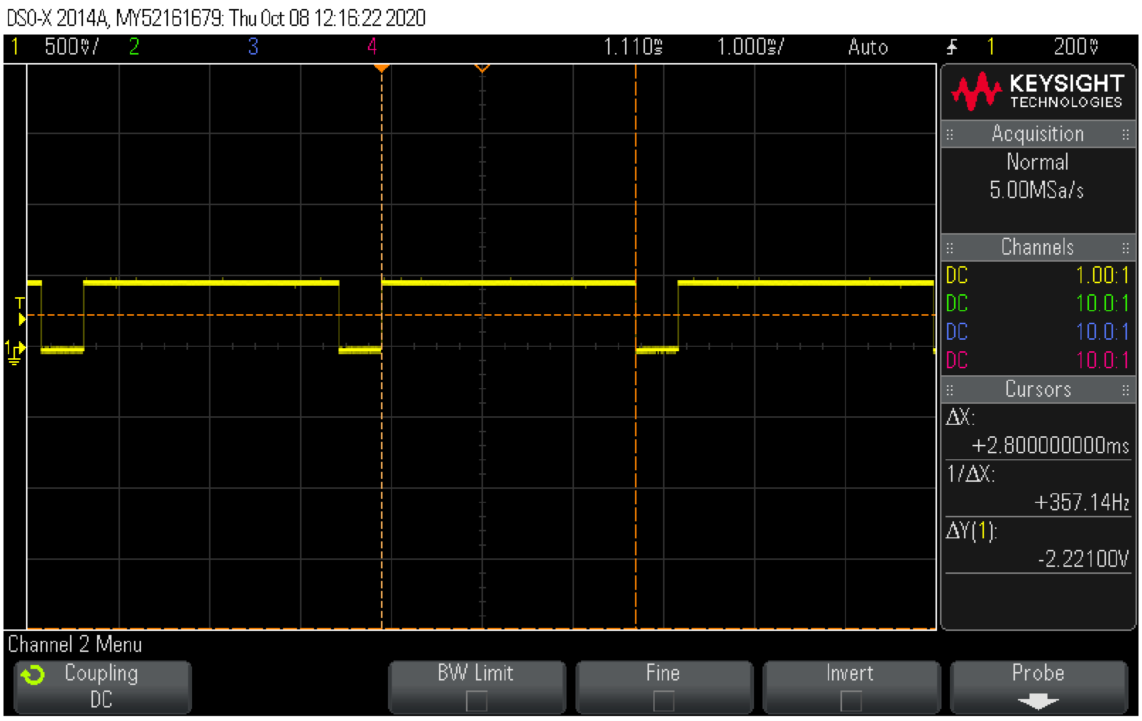


Figure 17: Echo pulse observed in Oscilloscope

When the echo pin was connected to the oscilloscope, the image as shown in figure 17 was obtained. Figure 17, shows the echo pulse going HIGH for 2.80 ms (ΔX) which means the object was at a distance of 48 cm.

In this thesis project, an ultrasonic sensor was used to detect the fluctuations in the surroundings when certain criteria were met which depends upon the application it is being used for. For example, if it is used in a parking assistance system it will measure the distance to the obstacle and when the obstacle starts to get too close to the vehicle at a certain distance it sends a signal to the buzzer and the buzzer starts beeping.

4.3 Humidity and Temperature Sensor (DHT22)

Grove - Humidity and Temperature Sensor (DHT22) is a multi-purpose sensor that can measure temperature (°C) and relative humidity (RH) at the same time. It provides a pre-calibrated digital output that can be directly sent to output devices like screens. To measure the humidity a capacitive sensor is used, whereas a negative temperature coefficient thermistor (NTC) is used to measure temperature.

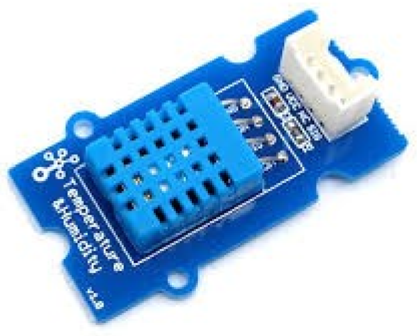


Figure 18: Grove- Humidity and Temperature Sensor (13)

As shown in figure 18, the DHT22 sensor has four pins, VCC and GND pins for power and signal pins for data transmission.

4.3.1 Capacitive Humidity Sensor

In capacitive humidity sensors, the change in capacitance is used to measure the humidity of the surrounding. These kinds of sensors are widely used, and they are easy to utilize for different purposes. The capacitive sensor consists of two metal plates separated by a thin layer of the non-conductive polymer film which acts as a dielectric portion of the capacitor (14).

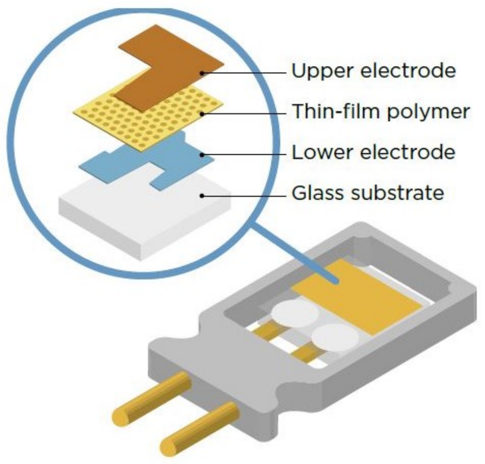


Figure 19: Structure of Capacitive Sensor. Reprinted from (14)

As shown in figure 19, the sensor consists of two electrodes and a thin-film polymer with a glass substrate at the base to support the sensor.

The upper electrode acts as a conductive material and one of the two sides of electrodes of the capacitor, which protects the active thin polymer from dust and dirt and allows the water vapor to get through. The active material, polymer film, absorbs water vapor from the surrounding. For the lower electrode, a conductive material is used which acts as one of the two electrodes in a capacitor.

The capacitance of a parallel plate capacitor is proportional to the area, A in m^2 of the smallest of the two plates and inversely proportional to the distance or separation, d (i.e. the dielectric thickness) given in meters between these two conductive plates (15).

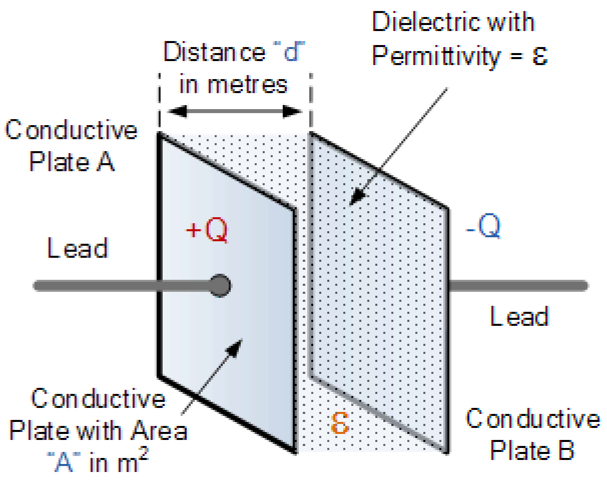


Figure 20: Inside a Capacitor. Reprinted from (15)

A capacitor has the components as shown in figure 20. In a sensor, both plates are fixed and the only thing that can be changed is dielectric permittivity which changes the capacitance of the capacitor. In a humidity sensor, the change in humidity changes the dielectric of the capacitor which affects the capacitance. From the experimental results, the capacitance was observed to increase with higher humidity, which demonstrates sensitivity to humidity in the studied range. Higher water molecule content at high humidity levels increases the dielectric constant, thereby acting to increase the capacitance of the sensor (16).

To calculate the capacitance of the capacitor, shown in figure 21, the following formula can be used.

$$\mathbf{C = \varepsilon \frac{A}{d}} \quad (4)$$

Alongside the two plates of the capacitor, the factor that affects the capacitance of the device is the dielectric material used (Permittivity). So, the actual permittivity between the two plates is the product of the permittivity of the free space(ε_o) and the relative permittivity of the material used(ε_r).

$$\boldsymbol{\varepsilon = \varepsilon o * \varepsilon r} \quad (5)$$

So, equation 4 becomes,

$$\boldsymbol{C} = \boldsymbol{\varepsilon o} * \boldsymbol{\varepsilon r}\frac{\boldsymbol{A}}{\boldsymbol{d}} \qquad (6)$$

The dielectric constant used for common materials are: Air =1.0006, Glass = 3 to 10, Vacuum = 1, Wood = 3 to 8 etc. The slight change in permittivity changes the capacitance of the capacitor. As the humidity changes the permittivity changes and as permittivity changes the capacitance of the capacitor changes.

4.3.2 Negative Temperature Coefficient Thermistor (NTC)

A thermistor is a type of resistor whose resistance depends upon the temperature. NTC thermistor, resistance decreases as temperature rises usually due to an increase in conduction electrons bumped up by thermal agitation from the valance band. An NTC is commonly used as a temperature sensor in different applications like HVAC, automotive, refrigeration applications, and medical applications (17).

Figure 21 shows the temperature and resistance curve on the left and NTC thermistor on the right.

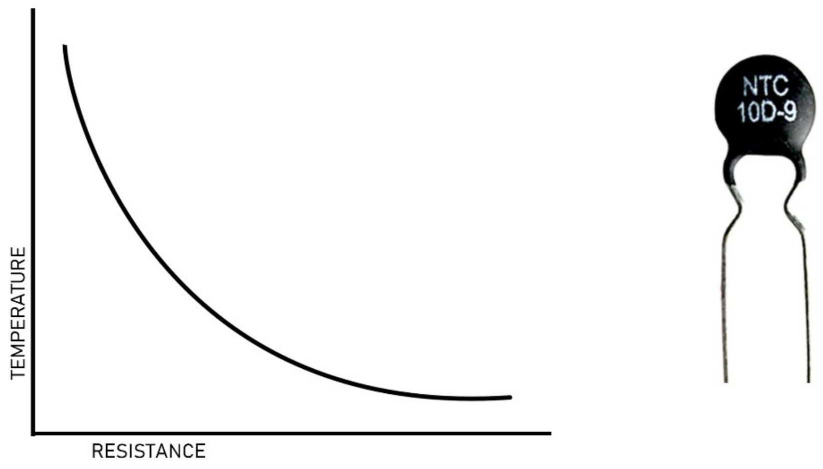


Figure 21: Temperature Vs Resistance (Left) NTC thermistor (Right) (18)

4.4 OLED

OLED stands for the organic light-emitting diode. OLED is a thin film of organic compound which in response to electric current emits light. In this project, a small OLED display of 0.96 inches is used to display the ultrasonic sensor values alongside humidity and temperature sensor values. It uses an I2C communication protocol to communicate with the microcontroller, so it is easy to use with the Photon Board.

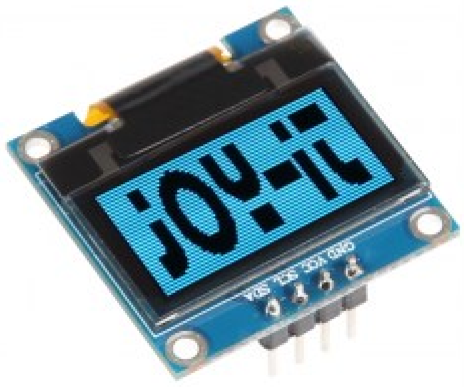


Figure 22: OLED Display Reprinted from (19)

As shown in figure 22, it consists of four pins SDA, SCL, GND, and VCC for the I2C communication.

4.5 Piezo Buzzer

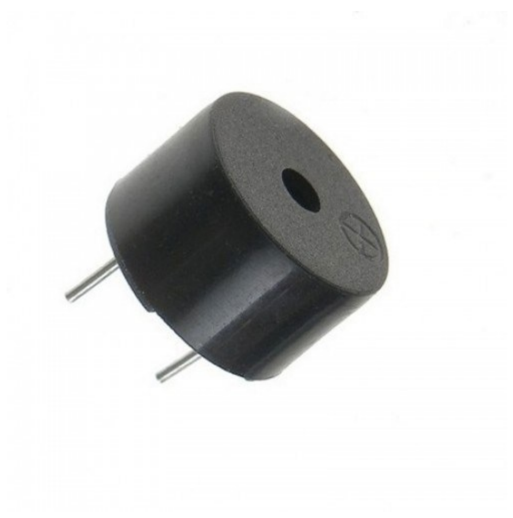


Figure 23: Piezo Buzzer. Reprinted from (20)

Piezo buzzer is a small electronic device to generate basic sound tones in beeps. They are small, low-cost, and can be mounted on printed circuit boards effortlessly. Piezo buzzers are commonly used in alarms, games, computer devices, automobiles, etc. These kinds of buzzers are generally two-legged with two terminals, positive and negative as shown in figure 23. It can work with low voltage, 3 to 5 volts, and draw a maximum current of 10 mA

The working component in most audible sound transducers is a thin disk of piezoelectric ceramic bonded to a similarly thin metal diaphragm. The ceramic disk deforms when voltage is applied and bends the metal diaphragm. The vibration of metal at a certain frequency produces audible sound (21). The pitch of the sound can be altered with the change in voltage.

In this project, the buzzer was used to generate beep sounds when certain conditions are met. For example, if the ultrasonic sensor picks up anything within its radar, the buzzer produces a sound to notify the user.

4.6 TTL Serial Camera

A TTL is a serial camera module commonly used to take a photo or to control a video stream with NTSC video output. It has a maximum image of 640x480 pixels with a CMOS sensor which produces JPEG and VGA output formats.

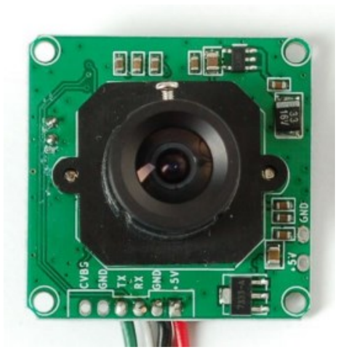


Figure 24: TTL Camera Module. Reprinted form (22)

This type of camera module has dimensions of 32 mm x 32 mm which is small as shown in figure 24. It has a two-wire communication with TX to transmit and RX to receive the data with DC +5 V operating voltages and a maximum current of 75 mA. In this project, a TTL camera is used to get the live feed of the surrounding in the local web host address.

5 System Design

All the above-explained sensors and devices are coalesced to design a smart system that can be used in different applications. A system layout is designed below to show the overview of the project.

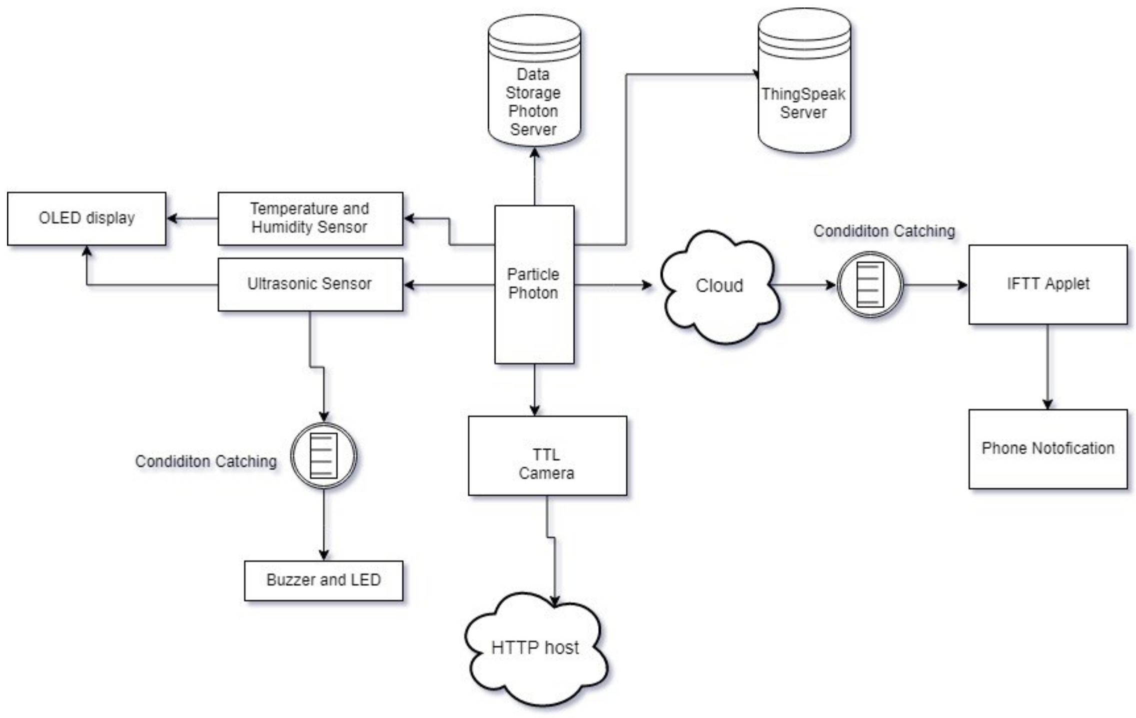


Figure 25: System Layout

As shown in figure 25, different kinds of sensors are used to interact with the photon. Ultrasonic sensor (HC-SR04) was used to measure the proximity of the object using ultrasound whereas humidity and temperature sensor (DHT-22) measures the humidity and temperature and the respective values were displayed using the OLED screen. Also, the data can be analyzed from an online platform called ThingSpeak, which can be used by creating a webhook in particle photon Web IDE. Particle photon acquires data and sounds a buzzer and lights up the LED to notify to changes which can set using the Particle Web IDE. Also, the IFTTT application sends a notification to the user's phone at certain conditions which are set in the IFTTT applet. Also, a TTL camera was added to get a live feed of the surrounding.

The system build in this project can be used in different applications like Parking Assistance, Anti-Theft Alarm, Automatic Water Leveling, and Monitoring System Analyze the Live Data and Control the Devices Wirelessly, and many more.

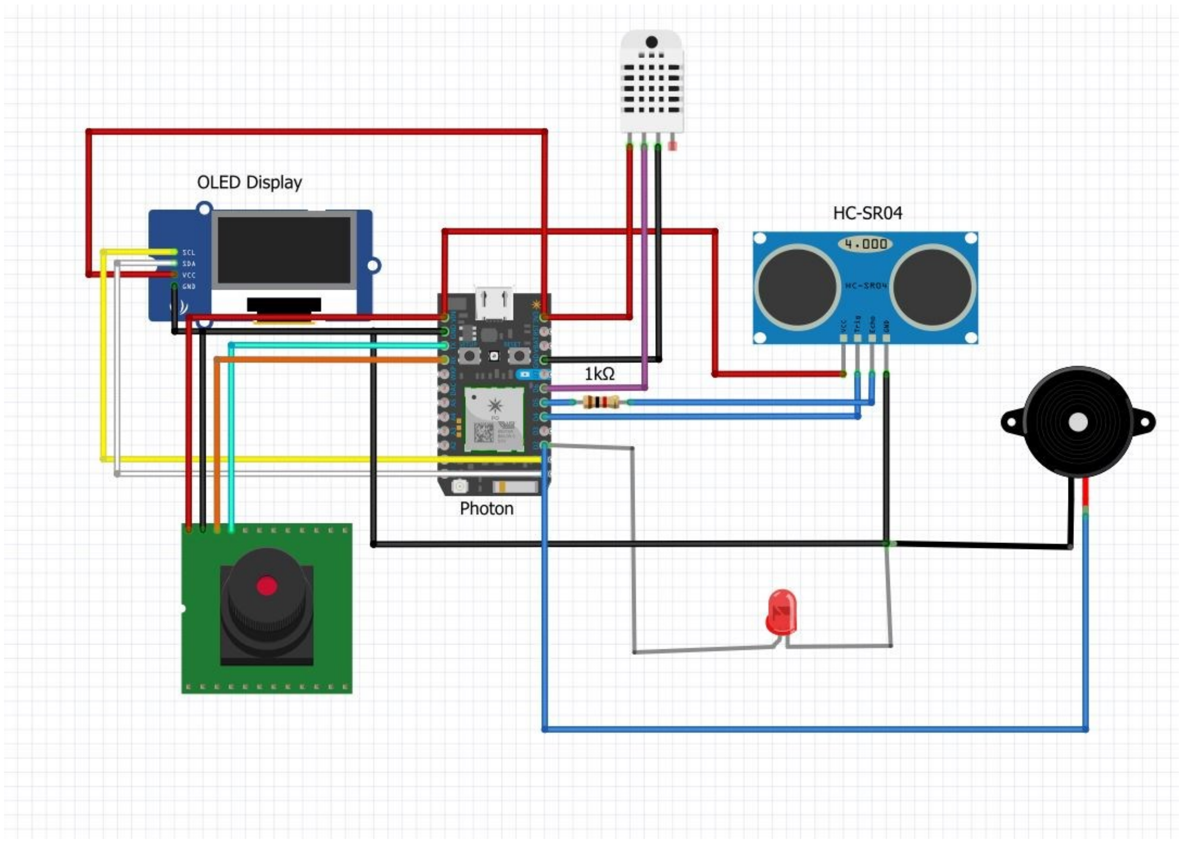


Figure 26: System Design

Figure 26 shows the final system design of the hardware and pins are connected accordingly. The ultrasonic sensor detects the changes and notifies the particle photon which then sends signals to the buzzer and LED whereas the OLED display shows the value of the ultrasonic sensor and humidity and temperature sensor. TTL camera captures the picture and post on the local address.

For ultrasonic sensor, *#include <HC-SR04.h>* library was used, for humidity and temperature sensor *#include <Adafruit_DHT.h>* library was used and for OLED display *#include <Adafruit_SSD1306.h>* library was used. The full working code is in Appendix 1-3.
For TTL camera separate code was written so that it can be run on a local machine. Node JS and C++ were used to program the TTL camera to work in localhost *http://localhost:3000/*. The full code for the TTL camera is in Github.

5.1 Creating Applets and Webhook for Data Analysis

IFTTT Applets

IFTTT applets can be created using the IFTTT platforms where sensor value changes are required for the trigger. To get the notifications on the phone IFTTT application was used which is available for both IOS and Android operating systems in the play store. *Particle.publish("Distance-CM", String(distCm));* were used to create an event as shown in Listing 1. These events publish the distance which triggers the event on the IFTTT application and sends the notification.

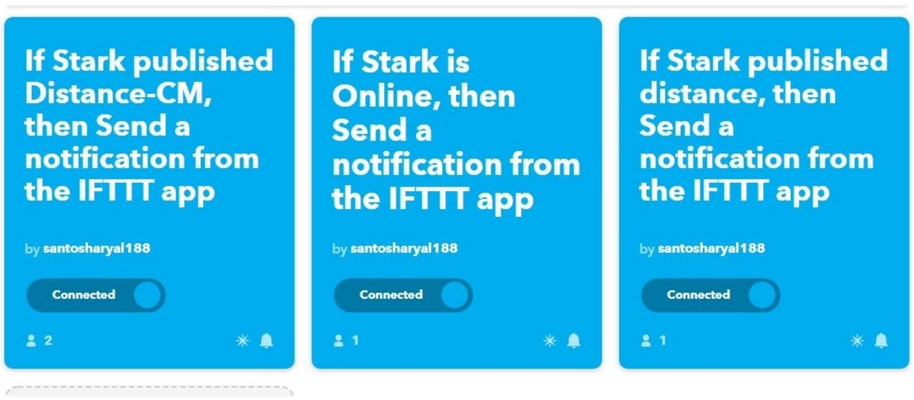
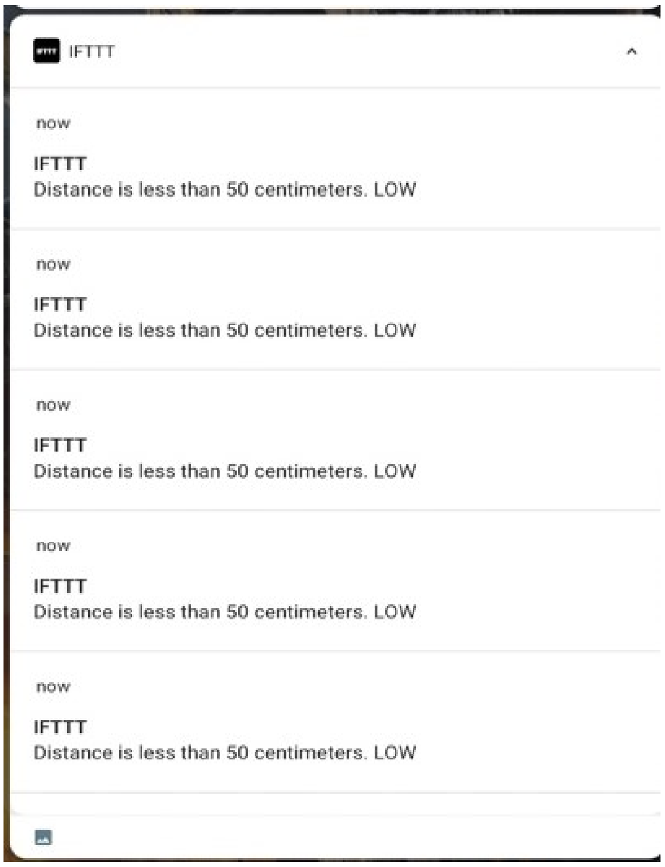

If Stark published Distance-CM, then Send a notification from the IFTTT app
by santosharyal188
Connected
2
If Stark is Online, then Send a notification from the IFTTT app
by santosharyal188
Connected
1
If Stark published distance, then Send a notification from the IFTTT app
by santosharyal188
Connected
1

Figure 27: IFTTT Applets

As shown in figure 27, when a particle photon named Stark receives the values from an ultrasonic sensor, the IFTTT applet triggers the event and sends the notification to the IFTTT application which is installed on the phone. For example, if the distance is less than 50 cm, it can send the notification, “Distance is less than 50 cm. LOW” which can be also seen in figure 28, to the IFTTT app using *Particle.publish("Distance-CM", "LOW");* event but if the distance is more than 50 cm, it can use *Particle.publish("Distance-CM","HIGH");* event to send the notification.

The same method can be used for humidity and temperature sensor to create a notification when the temperature or humidity changes.



IFTTT

now

IFTTT
Distance is less than 50 centimeters. LOW

now

IFTTT
Distance is less than 50 centimeters. LOW

now

IFTTT
Distance is less than 50 centimeters. LOW

now

IFTTT
Distance is less than 50 centimeters. LOW

now

IFTTT
Distance is less than 50 centimeters. LOW

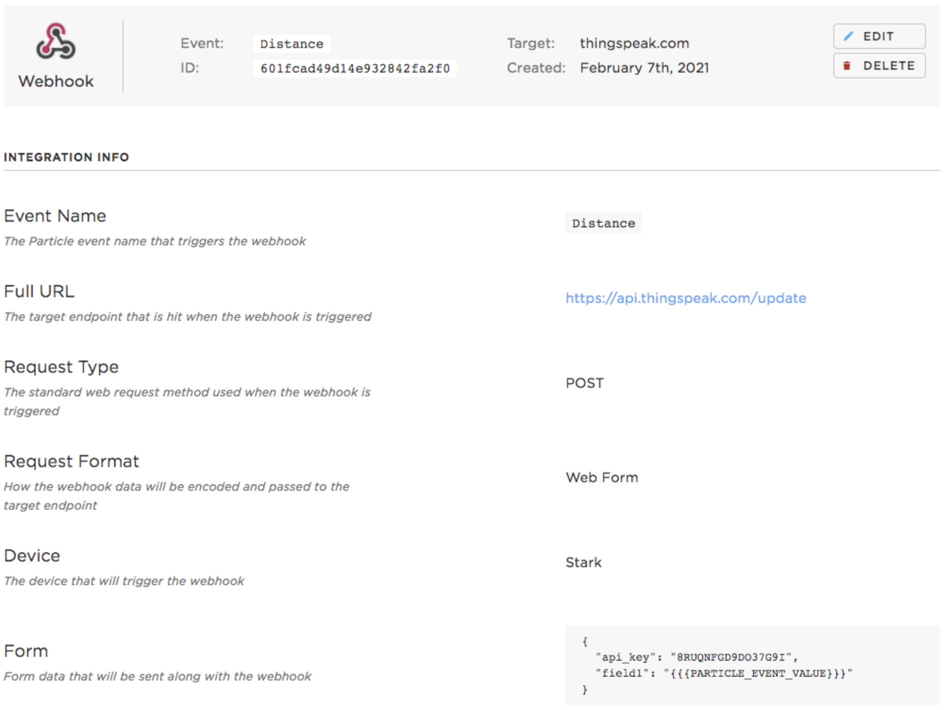
Figure 28: Phone notification

Figure 28 shows the notification sent by the IFTTT application to the phone when the event is triggered.

MATLAB Web-Hook

A webhook provides third-party applications with real-time information. As soon as the sensor detects the changes, an event is triggered, the sensor sends the data to the webhook and the user can get the data immediately.

Photon IDE provides a platform to create Webhooks. To create a Webhook, the integration tab on the Photon Web IDE can be used where it provides the Webhook builder. In this project, Matlab Web-hook was used for data analysis for the ultrasonic sensor.



Webhook

Event: Distance

ID: 601fcad49d14e932842fa2f0

Target: thingspeak.com

Created: February 7th, 2021

EDIT

DELETE

INTEGRATION INFO

Event Name *The Particle event name that triggers the webhook*	Distance
Full URL *The target endpoint that is hit when the webhook is triggered*	https://api.thingspeak.com/update
Request Type *The standard web request method used when the webhook is triggered*	POST
Request Format *How the webhook data will be encoded and passed to the target endpoint*	Web Form
Device *The device that will trigger the webhook*	Stark
Form *Form data that will be sent along with the webhook*	

```
{
  "api_key": "8RUQNFGD9DO37G9I",
  "field1": "{{{PARTICLE_EVENT_VALUE}}}"
}
```

Figure 29: Webhook created using Particle IDE

Figure 29 shows the particle webhook created using particle IDE which sends the distance measurement to thingspeak server. In the webhook, an event name is required to get the desired data. Thingspeak provides the API key to acquire the data from the Particle server which you can see in figure 30, in the Form section.

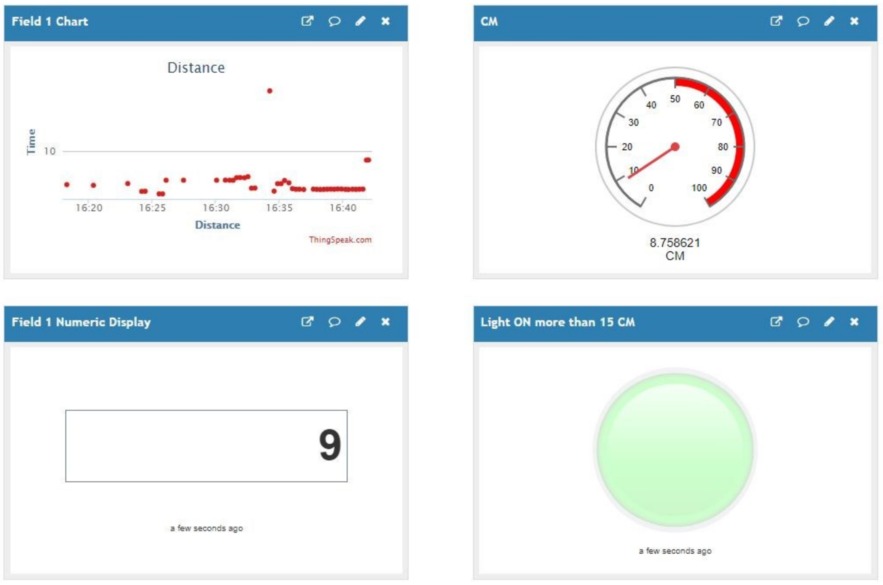


Figure 30: Thingspeak Channel

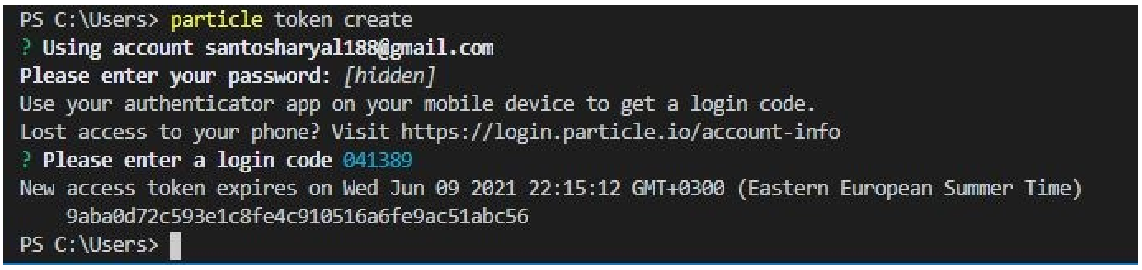
Figure 30 shows the thingspeak channels and the data received for the particle server over time. Any kind of data can be stored in this server for future use and analysis.

5.2 TTL Camera Setup

5.2.1 Running on Local Machine

Visual Studio Code is used to create a local machine and execute the code for the TTL camera. It also provides a command-line interface (CLI) to interact with the particle photon whenever needed.

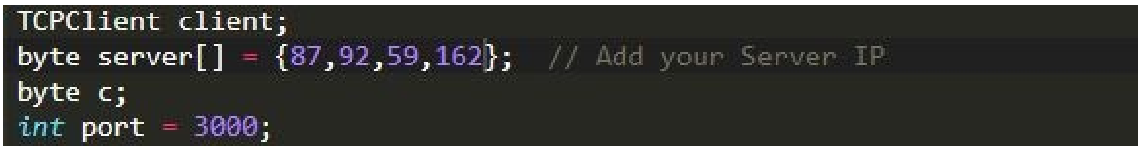
For the camera to send data to localhost, a website was created using Node JS that runs in the local machine. To connect the camera to the website, an access token provided by the particle photon was used alongside the local server IP address. For the camera functions, the library *#define VC0706* was used.



```
PS C:\Users> particle token create
? Using account santosharyal188@gmail.com
Please enter your password: [hidden]
Use your authenticator app on your mobile device to get a login code.
Lost access to your phone? Visit https://login.particle.io/account-info
? Please enter a login code 041389
New access token expires on Wed Jun 09 2021 22:15:12 GMT+0300 (Eastern European Summer Time)
    9aba0d72c593e1c8fe4c910516a6fe9ac51abc56
PS C:\Users>
```

Figure 31: Getting Particle Token Using VS Code CLI

VS code CLI was used to get the access token for the particle photon as shown in figure 31. Particle provides a new access token every time particle token is created which lasts for three months by default.



```
TCPClient client;
byte server[] = {87,92,59,162};  // Add your Server IP
byte c;
int port = 3000;
```

Figure 32: IP SERVER and Port

The code for the camera can be found in https://github.com/IronmanMark2/ParticlePhotonTTLCamera with comments.

After the camera was connected as shown in figure 26, the byte server and the port were changed in the code *ttlcameraparticlephoton.ino* as shown in figure 32. Then the code was flashed in the firmware of the particle photon.

To create a web host, code was written in Node JS, and Visual Studio Code was used to run it. The code can be found in https://github.com/IronmanMark2/WebTTLcamera with comments. All the files were downloaded, and they were put together in one folder named ttlcamera which was in c drive of the computer. Then visual studio CLI was used to access the folder using the cd command, *cd C:\ttlcamera.*

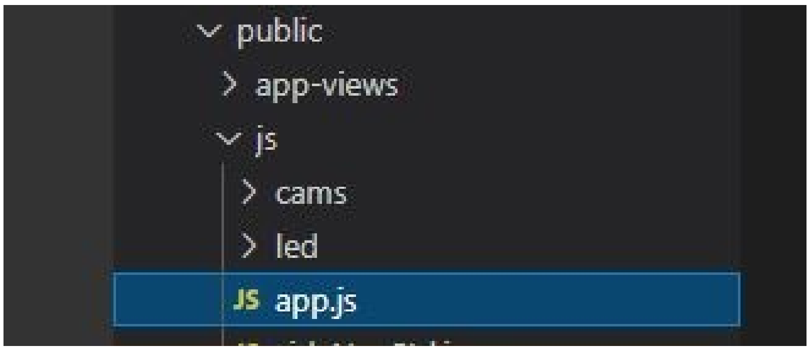
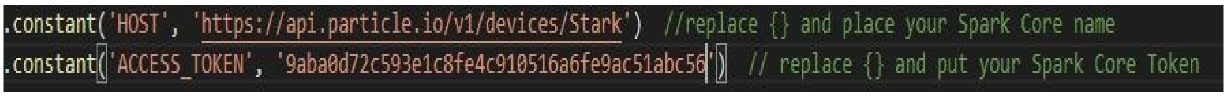


Figure 33: Folder that contains app.js



```
.constant('HOST', 'https://api.particle.io/v1/devices/Stark')  //replace {} and place your Spark Core name
.constant('ACCESS_TOKEN', '9aba0d72c593e1c8fe4c910516a6fe9ac51abc56')  // replace {} and put your Spark Core Token
```

Figure 34: Changing access token

In the folder ttlcamera*/public/js/app.js* as shown in figure 33, access token and hostname were changed as shown in figure 34.

To run the Node JS, *npm* package manager was installed using the command *npm install*, which provides the hosting for software development and is mostly used to publish, discover, install and develop node programs.

To run a local server npm package was initialized by *npm start* which created a website in local server as shown in figure 35, which can be accessed http://localhost:3000 address.

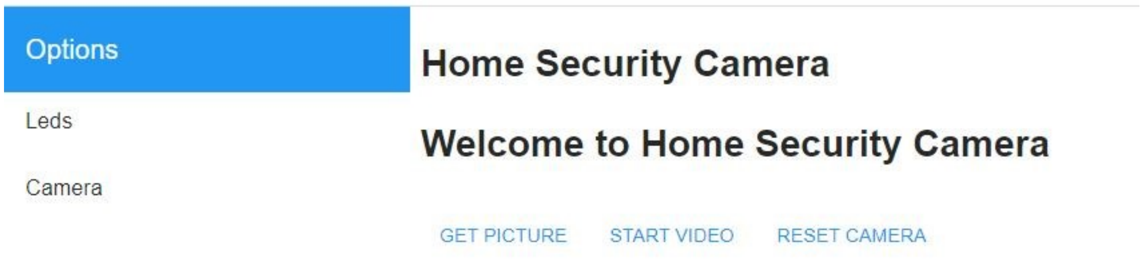


Figure 35: TTL Camera Interface

Figure 35 shows the camera interface which runs in the local machine which has a function like a get picture, start video, and reset the camera.

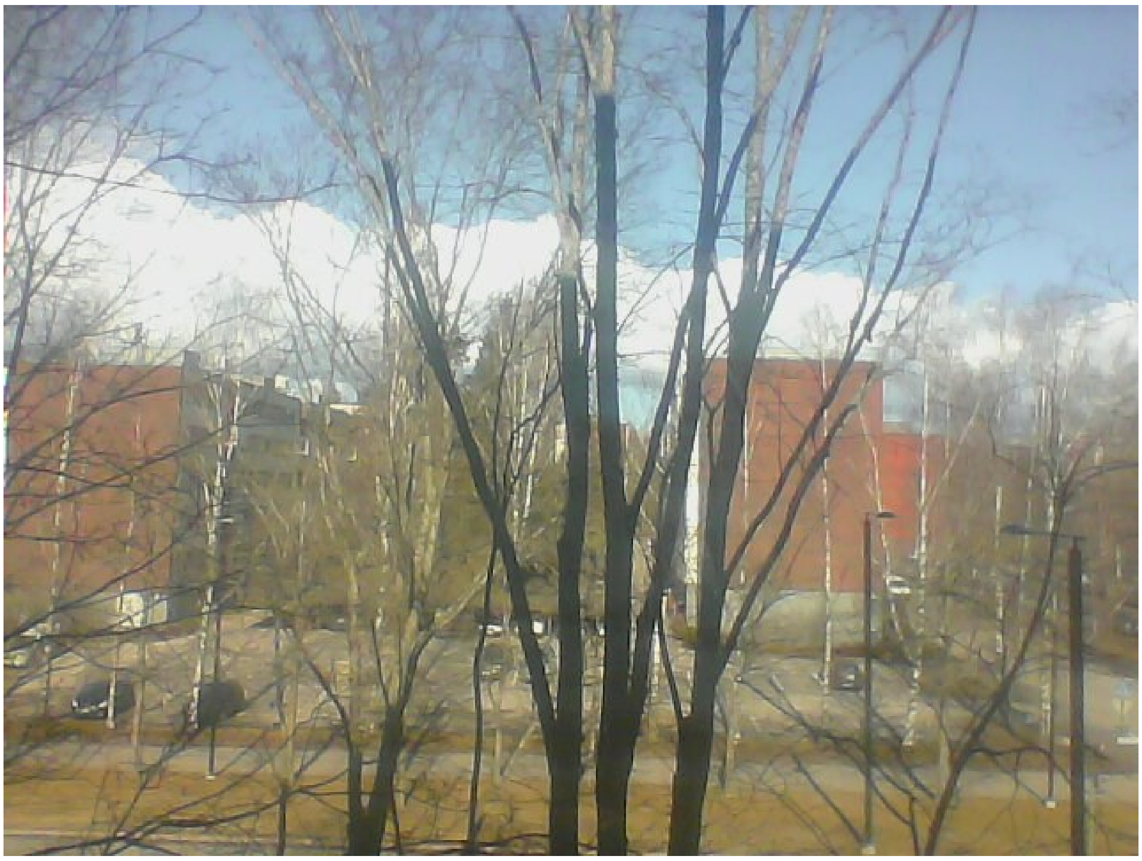


Figure 36: Picture taken from TTL Camera

Figure 36 shows the picture of the surroundings taken from the TTL camera which is 640x480 pixels size.

6 Conclusion

In this project, the goal to build a smart system that can be used in different applications was achieved. Various sensors alongside powerful particle photon boards were used. An ultrasonic sensor was used to measure the proximity of the object using ultrasound, whereas humidity and temperature sensor measures the humidity and temperature which was displayed in the OLED display. A buzzer was used to notify the changes and with the help of the camera, the live view of the surroundings can be seen. Also using an IFTT platform, notification was sent to the user's phone if any changes occur. The accumulated data can be analyzed from anywhere using ThingSpeak platform.

The system built in this project can be used in various applications in the modern-day world. Antitheft System, Parking Assistance, Water Level monitoring in the water reservoir, and Data analysis are the most common areas to use this kind of smart system. Camera, ultrasonic sensor, and buzzer are useful in parking assistance systems in an automobile. When notification to user's phone feature is added, the system becomes more beneficial in antitheft alarm system and water level monitoring system. In this system, data can be sent through webhooks which is useful in data analysis and data storage for future use.

While testing the system the results were promising can be further developed to make it better. This system can be further developed and designed in printed circuit boards where components can be mounted. Also, a power source can be added to the system which makes it portable.

While building this project the coding part for the TTL camera was the most challenging. Since it deals with the wireless transmission of data to the localhost there was more software than hardware for this segment of the project. All the code for the TTL camera was posted in GITHUB because it was too large to include in this document.

References

1. **Particle Industries, Inc.** Particle IoT. *Particle.* [Online] 2021. [Cited: 01 14, 2021.] https://www.particle.io/iot-platform.

2. **Particle docs.** Photon Datasheet. [Online] [Cited: July 25, 2020.] https://docs.particle.io/datasheets/wi-fi/photon-datasheet/.

3. **Particle docs.** Flash APPS WITH THE PARTICLE WEB IDE. [Online] Particle. [Cited: July 28, 2020.] https://docs.particle.io/tutorials/developer-tools/build/.

4. **Particle docs.** PARTICLE CLI. [Online] Particle. [Cited: July 28, 2020.] https://docs.particle.io/tutorials/developer-tools/cli/.

5. **Particle docs.** TINKERING WITH "TINKER"-PHTOTN. [Online] [Cited: August 28, 2020.] https://docs.particle.io/tutorials/developer-tools/tinker/photon/.

6. **Afzal, Sal.** I2C Primer. *ANALOG DEVICES.* [Online] [Cited: September 2, 2020.] https://www.analog.com/en/technical-articles/i2c-primer-what-is-i2c-part-1.html.

7. **TEXAS INSTRUMENTS.** Understanding the I2C Bus. [Online] [Cited: September 6, 2020.] https://www.ti.com/lit/an/slva704/slva704.pdf?ts=1603879247666&ref_url=https%253A%252F%252Fwww.google.com%252F.

8. **Wikipedia Org.** Serial Peripheral Interface. [Online] September 3, 2020. [Cited: September 8, 2020.] https://en.wikipedia.org/wiki/Serial_Peripheral_Interface.

9. **Texas Instruments.** *TMS320DM646x DMSoS Serial Peripheral Interface (SPI).* Texas : Texas Instruments, 2011. p. 45, User's Guide. SPRUER4B.

10. **JHDLCD.** SPECIFICATION OF LCD MODULE. [Online] 1.1, JHDLCD, September 13, 2018. [Cited: September 28, 2020.] https://files.seeedstudio.com/wiki/Grove-16x2_LCD_Series/res/JDH_1804_Datasheet.pdf.

11. **Wikipedia Org.** Ultrasound. [Online] September 2020. [Cited: September 12, 2020.] https://en.wikipedia.org/wiki/Ultrasound.

12. **MANDU.** ULTRASONIC SENSOR - HC-SR04. *MANDU.* [Online] [Cited: 11 22, 2020.] https://mandu.fi/products/ultrasonic-sensor-hc-sr04.

13. **seeed.** Grove - Temperature&Humidity Sensor (DHT11). *seeed the Iot hardware enabler.* [Online] [Cited: 04 07, 2021 .]

14. **Bennett-Livingston, Janice.** Humidity sensors – do you know what's inside? [Online] VAISALA, May 14, 2015. [Cited: 11 18, 2020.] https://www.vaisala.com/en/blog/2020-10/humidity-sensors-do-you-know-whats-inside.

15. **Electronics Tutorials.** Introduction to Capacitors. [Online] [Cited: 11 29, 2020.] https://www.electronics-tutorials.ws/capacitor/cap_1.html.

16. **Hassan Maktuff Jaber Al-Ta, Yusoff Mohd Amin and Vengadesh Periasamy.** *Humidity influenced capacitance and resistance of an Al/DNA/Al Schottky diode irradiated by alpha particles.* s.l. : scientific reports, 2016.

17. **Wikipedia Org.** Thermistor. [Online] [Cited: 12 5, 2020.] https://en.wikipedia.org/wiki/Thermistor.

18. **Ametherm.** NTC Thermistor Beta. *Ametherm.* [Online] [Cited: 01 05, 2021.] https://www.ametherm.com/thermistor/ntc-thermistor-beta.

19. **Partco.** Joy-IT OLED. [Online] [Cited: 01 11, 2021.] https://www.partco.fi/fi/naeytoet/oled-naeytoet/21103-sbc-oled01.html.

20. **Pcboard.** Mini Piezo Buzzer Mini PCB Mount. *Pcboard.ca.* [Online] [Cited: 01 18, 2021.] https://www.pcboard.ca/minipiezo-buzzer.

21. **APC International, Ltd.** PIEZO BUZZERS. [Online] [Cited: 01 07, 2021.] https://www.americanpiezo.com/standard-products/buzzers.html.

22. **Ada, Lady.** TTL Serail Camera. [Online] 10 12, 2018. [Cited: 01 11, 2021.] https://www.mouser.fi/datasheet/2/737/ttl-serial-camera-932886.pdf.

23. **Pedamkar, Priya.** What is IOT? *EDUCBA.* [Online] [Cited: 04 11, 2021.] https://www.educba.com/what-is-iot/.

24. **Wikipedia Org.** Speed of Sound. *Wikipedia The Free Encyclopedia.* [Online] [Cited: September 25, 2020.] https://en.wikipedia.org/wiki/Speed_of_sound.

Code for Humidity and Temperature Sensor, OLED display, and Ultrasonic Sensor

```
#include <Adafruit_SSD1306.h>
#include <Adafruit_DHT.h>
#include <HC-SR04.h>
#include <HC-SR04.h>
#if defined (PARTICLE)
#else
#include <Wire.h>
#endif
#define DHTPIN D6
#define DHTTYPE DHT11
DHT dht(DHTPIN,DHTTYPE);
#define OLED_RESET D4
Adafruit_SSD1306 oled(OLED_RESET);
int trigPin = D4;
int echoPin = D5;
int pin=D2;
int maxDist = 200;
int minDist = 0;
float distCm = 0.0;
double getDistanceCM()
{
   sendTriggerPulse(trigPin);
   waitForEcho(echoPin, HIGH, 100);
   long startTime = micros();
   waitForEcho(echoPin, LOW, 100);
   long endTime = micros();
   long duration = endTime - startTime;
   double distance = duration / 29.0 / 2.0;
   if (distance < minDist || distance > maxDist)
   {
      return -1;
   }
   return distance;
}
void sendTriggerPulse(int pin)
{
   digitalWrite(pin, HIGH);
   delayMicroseconds(10);
   digitalWrite(pin, LOW);
}
void waitForEcho(int pin, int value, long timeout)
{
   long giveupTime = millis() + timeout;
   while (digitalRead(pin) != value && millis() < giveupTime)
   {
   }
}

void setup()
{
   Particle.publish("State","DHT22 test start");
   dht.begin();
   oled.begin(SSD1306_SWITCHCAPVCC, 0x3C);
   delay(1000);

   pinMode(pin,OUTPUT);
   pinMode(trigPin, OUTPUT);
   pinMode(echoPin, INPUT);
   delay(1000);
}
void loop()
{
   float h = 0.0;
   float t = 0.0;


   h=dht.getHumidity();
   t=dht.getTempCelcius();
   delay(5000);

char finalPrint[100] ;
 sprintf(finalPrint , "Humi: %3.2f %c\nTemp: %3.2f%cC \nDistance:  %3.2f %c" , h ,'%', t , ' ' ,distCm) ;
oled.clearDisplay();
delay(1000);
oled.setTextSize(1.8);
oled.setTextColor(WHITE);
oled.setCursor(0,0);
oled.print( finalPrint );
oled.setTextColor(BLACK, WHITE);
oled.display();
delay(1000);
Particle.publish("Humidity", String(h) + "%");
Particle.publish("Temperature", String(t) + " °C");
//Ultrasonic sensor
   distCm = getDistanceCM();
   Particle.publish("Distance-CM", String(distCm));
   if(distCm>=40){
      Particle.publish("Distance-CM","HIGH");
      digitalWrite(pin,HIGH);
      delay(1000);
      digitalWrite(pin,LOW);
        delay(1000);
   }
   else {
      Particle.publish("Distance-CM","LOW");
    delay(1000);
}
```