A Report on

Automated Overhead Tank Water Level Control System

For

Mini Project 1-b (REV- 2019 ‘C’ Scheme) of Second Year, (SE Sem-IV)

In

Electronics & Telecommunication Engineering

By

1.	Trisha Gaur	(15)
2.	Mcvean Soans	(34)
3.	Sharvani Sawant	(47)
4.	Shubham Sangani	(63)

Under the guidance of

Mr. Mrugendra Vasmatkar

Department of Electronics and Telecommunication
Vivekanand Education Society’s Institute of Technology
UNIVERSITY OF MUMBAI
Academic Year 2020-21

CERTIFICATE

This is to certify that the project entitled **Automated Overhead Tank Water Level Control System** is a bonafide work of

1. Trisha Gaur
2. Mcvean Soans
3. Sharvani Sawant
4. Shubham Sangani

submitted to the University of Mumbai in partial fulfillment of the requirement for the award of **Mini Project 1-b (REV- 2019 'C' Scheme) of Second Year, (SE Sem-IV)** in **Electronics & Telecommunication Engineering** as laid down by **University of Mumbai** during academic year **2020-21**

(______________________)	(______________________)
Examiner/Reviewer-1	**Examiner/ Reviewer -2**

Mr. Mrugendra Vasmatkar	**Mrs. Shoba Krishnan**	**Dr. J.M. Nair**
Project Guide	**Head of Department**	**Principal**

ABSTRACT

Water shortage has been an important issue faced by the people living in Societies not only in Mumbai, but also other places within the state of Maharashtra. Tackling the issue of water shortage is of utmost importance. Many factors contribute to water scarcity, the most prominent one being taps left open and unattended. In large housing societies, the Security or Watchman has to be alert and keep an eye on the overhead tanks of several buildings, and not only this, the person is required to physically go to the location of the overhead tanks (usually rooftops of the buildings) in order to control the valves that supply water. It is a pretty hectic task and to overcome this problem, the necessity of an Automated Centralized System was identified.

To deal with the water wastage issues faced by Co-operative Housing Societies, an automated system is a must to deal with problems including Speed, Efficiency and any other previously Unforeseen Situations. The proposed partially-wireless system using NodeMCU which employs an IOT-based architecture to facilitate remote working. The system will be used not only by the flat owners, but also by the Society personnel, thereby further reducing the possibility of wastage and increasing the efficiency of the automated system. More so, the entire system will be centralized in order to simplify monitoring and control of water supply to the individual flats in various buildings.

A Centralized Partially-Wireless System has been developed to counter the issue of water wastage in many Societies. Hardware Implementation of the NodeMCU - based architecture and the various components associated with it has been completed. The assembly simulation using the user's device (utilizing the Blynk IoT Platform) has also been completed.

INDEX

ABBREVIATIONS

IDE	Integrated Development Environment	6
SPDT	Single Pole Double Throw	6
NodeMCU	Node Microcontroller Unit	6
IoT	Internet of Things	6
LAN	Local Area Network	7
GB	Giga Byte	7
RAM	Random Access Memory	7
MQTT	Message Queuing Telemetry Transport	9
GPIO	General Purpose Input / Output	11
LCD	Liquid Crystal Display	12
I2C	Inter - Integrated Circuit	12
LED	Light Emitting Diode	12
USB	Universal Serial Bus	13
GND	Ground	15
PWM	Pulse Width Modulation	15
SDA	Serial Data	16
SCL	Serial Clock	16
TRIG	Trigger	18
PIN	Personal Identification Number	19
UI	User Interface	21

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 NEED

Water shortage has been an important issue faced by the people living in Societies not only in Mumbai, but also other places within the state of Maharashtra. Water shortage issues even contribute towards inflation in the cost of water supplied to people. In 2019, the Brihanmumbai Municipal Corporation (BMC) increased the water charges by 2.48%. This was in immediate succession to the previous year's (2018's) hike of 3.72%. Tackling the issue of water shortage is of utmost importance. Many factors contribute to water scarcity, the most prominent one being taps left open and unattended. In large housing societies, the Security or Watchman has to be alert and keep an eye on the overhead tanks of several buildings, and not only this, the person is required to physically go to the location of the overhead tanks (usually rooftops of the buildings) in order to control the valves that supply water. It is a pretty hectic task and to overcome this problem, the necessity of an Automated Centralized System was identified.

1.2 DEFINITION

Automation has transformed almost every industry, and has been adopted into various scenarios encountered in life. The key to successful automation or (implementing an automated system) is careful analysis of the described problem statement. The prerequisites to implementing an automated system must be carefully studied, in addition to the advantages and disadvantages of the utilities used while implementing the solution. This process, usually referred to as **Ideation,** often consumes the most amount of time. It is generally followed by the **Design** of the automated system and finally the **Implementation** of the proposed automated system. To deal with the water wastage issues faced by Co-operative Housing Societies, an automated system is a must to deal with problems including Speed, Efficiency and any other previously Unforeseen Situations.

The proposed partially-wireless system which employs an IOT-based architecture to facilitate remote working. The system will be used not only by the flat owners, but also by the Society personnel, thereby further reducing the possibility of wastage and increasing the efficiency of the automated system. More so, the entire system will be centralized in order to simplify monitoring and control of water supply to the individual flats in various buildings.

2. LITERATURE REVIEW

- A **Water Meter** was initially considered as a preventive measure against overflow situations. It measures the volume of water which passes through an outlet (or pipe), thereby allowing us to measure the rate of water being supplied to a particular flat. It was discarded due to high cost and maintenance, leading towards an inefficient system. [8]
- We decided to utilize an **Ultrasonic Sensor** to determine the water level of the tank. It measures accurate distance using a non-contact technology (no physical contact between sensor and object). The depleting water level (indicating overflow) is quickly detectable, as the tank is continuously monitored. [12]
- A **2-channel Relay module** acts as a switch which connects the Solenoid Valve and the NodeMCU, enabling the control of current towards the valves. An SPDT relay consists of 2 circuits / parts - Electromagnetic and Mechanical. The Electromagnetic part consists of a coil which when energized sets up a temporary magnetic field responsible for the switching action that takes place in the Mechanical part of the device. Upon energizing the coil by a battery, the switch allows the current to flow from the battery to the Solenoid Valve, allowing for remote control of the flow of water. [7]
- A **Solenoid Valve** was chosen so as to allow remote, efficient, precise and speedy control of the flow of water towards flats. The solenoid coil present in the device, once energized, pushes the plunger outwards and stops the flow of water. The flow can be resumed upon de-energizing the solenoid coil. [6]
- **NodeMCU (ESP8266)** is an open-source Lua based firmware and development board specially targeted for IoT based Applications. It is one of the cheapest solutions on the market for DIY IoT and Home Automation projects. The ESP8266 NodeMCU can be programmed using the Arduino IDE programming environment. The NodeMCU serves as the core of our architecture which enables the partially-wireless system to control the water flow. [4][10][15]
- **Blynk** is a cloud-based architecture useful in creating IoT based systems. It consists of two essential components:

    - ➔ **Blynk App** - allows you to create amazing interfaces for your projects using various widgets we provide.
    - ➔ **Blynk Server** - responsible for all the communications between the smartphone and hardware. You can use our Blynk Cloud or run your private Blynk server locally. It’s open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi. [3]

- The user will be notified of a condition where they might not be able to control the water flow. We considered various variants of the Raspberry Pi. The Raspberry Pi 3 Model B is a single-board computer with wireless LAN and Bluetooth connectivity. The Raspberry Pi 4 sports a faster 1.5GHz clock speed processor (up from the 1.4GHz found on Raspberry Pi 3B+), with 1GB, 2GB and 4GB RAM variants. Raspberry Pi Zero is a mini single-chip computer. Finally, it was decided to use **Raspberry Pi Zero**, which is the smallest chipset in the Raspberry Pi series and is 40% faster than the original Raspberry Pi being nearly half of its size. [9]
- Softwares including LTSpice, Fritzing, Tinkercad, Proteus, Arduino IDE were looked at. We decided to utilize **Proteus** to design the Raspberry Pi-based architecture, as well as the NodeMCU-based architecture. The bit file generated using **Arduino IDE** was used to simulate the NodeMCU-based architecture circuit on Proteus. The block diagrams were created using a cloud-based design software called **Canva**. It was used as it promotes remote working and collaborations. [11][2]

3. PROBLEM STATEMENT

A major issue faced by housing societies in this day and age is the wastage of water. It occurs especially due to having water taps been left open and unattended during which water is unnecessarily squandered. The only way to prevent such wastage is by manual intervention. The drawback to this approach is that the concerned person has to be alert at all times in order to prevent excess wastage of water. If he is out and unable to switch off the supply, then there is no way this wastage can be prevented.

The proposed retort being an automated, centralized and partially-wireless system which can be operated by the user, as well as the society. It should be able to detect overflow situations and prevent excess water wastage. The flat owner as well as the society must be notified of such a situation, and a partially wireless control is to be given so as to prevent further wastage of water. The problem statement takes into account a particular flat may or may not have a tank placed within it, hence the system is required to cater towards both scenarios.

4. MINI PROJECT DESIGN

4.1 BLOCK DIAGRAM & DESCRIPTION

4.1.1 Raspberry Pi Implementation:

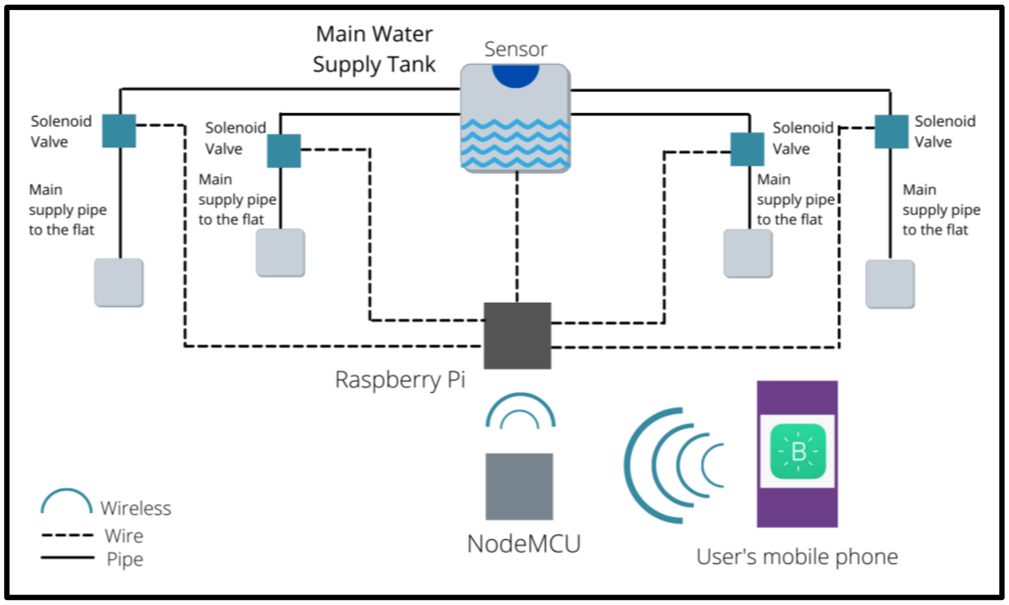


Figure 4.1. 1 Block Diagram (Raspberry Pi Implementation)

The Ultrasonic Sensor (primary **Sensing Element**) present within the Main Water Supply Tank, indicates the water level to the concerned authority. The **Solenoid Valves** connected to the Relay Module are used to control the flow of water, which is directed away from the Main Tank. The Sensing Element along with the Solenoid Valves comprise the **Mechanical Assembly**, which is connected to the **Raspberry Pi**. The Raspberry Pi device utilizes the MQTT Protocol and acts as a server to connect with the **NodeMCU** board. The mobile phone of the concerned authority (user), connected to the NodeMCU utilizing the **Blynk** software, acts as the Controlling Panel upon which we can monitor the water level of the Main Tank, and control the working of the valves which allow / restrict the water flow. The Blynk software is a Cloud-based IoT Platform, which provides a cloud server to communicate with hardware components, in this case, the Mechanical Assembly.

4.1.2 Only NodeMCU Implementation:

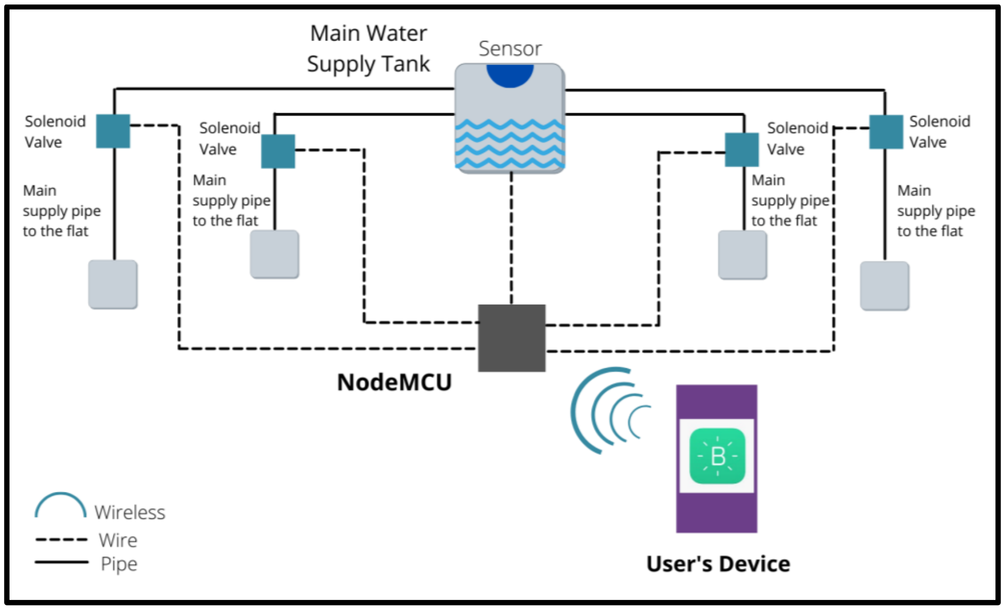


Figure 4.1. 2 Block Diagram (Only NodeMCU Implementation)

The Ultrasonic Sensor (primary **Sensing Element**) present within the Main Water Supply Tank, indicates the water level to the concerned authority. The **Solenoid Valves** connected to the Relay Module are used to control the flow of water, which is directed away from the Main Tank. The Sensing Element along with the Solenoid Valves comprise the **Mechanical Assembly**, which is connected to the **NodeMCU**. The mobile phone of the concerned authority (user) acts as the Controlling Panel upon which we can monitor the water level of the Main Tank, and control the working of the valves which allow / restrict the water flow. The system is designed as a safety measure against overflow situations, and a brisk response could be initiated using the application installed within the user’s device (mobile phone) to prevent excess water wastage. The user’s device is connected to the Mechanical Assembly via the NodeMCU utilizing a wireless means of communication between the same. The software used utilizes a Cloud-based IoT Platform called **Blynk**, which provides a cloud server to communicate with hardware components, in this case, the Mechanical Assembly.

4.2 CIRCUIT DIAGRAM & WORKING

4.2.1 Raspberry Pi Implementation

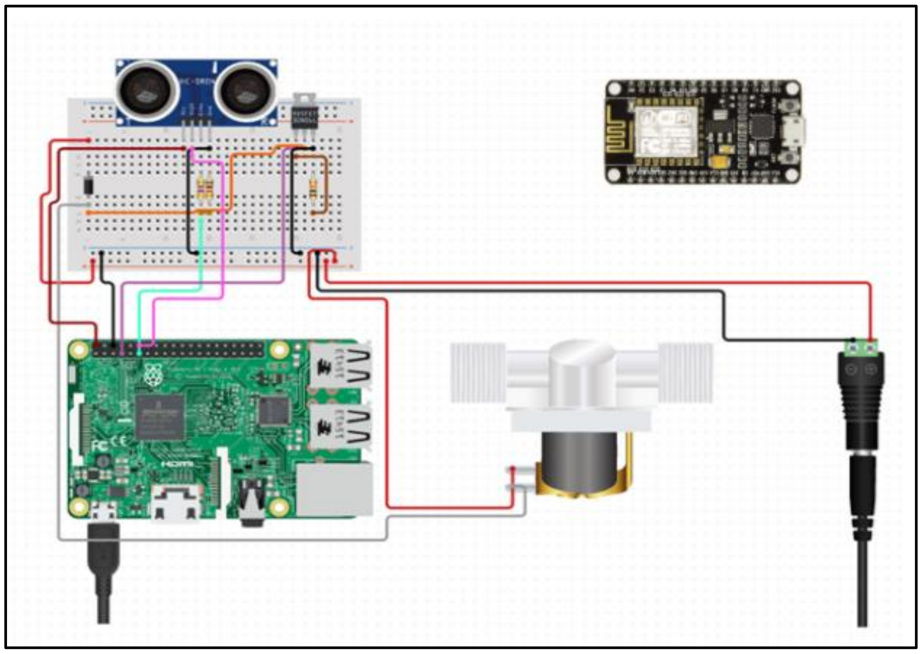


Figure 4.2. 1 Circuit Diagram of the System with Raspberry Pi

1. The Ultrasonic Sensor placed within the tank acts as the main sensing element of the system. The water level determined by the sensor is visible on the user's device.
2. The Solenoid Valve is responsible to control the water flow from the tank. The positive terminal is connected to a GPIO pin for a trigger input. Based on this trigger, the plunger (within the solenoid valve) opens / closes.
3. The Raspberry Pi is connected to the NodeMCU (ESP8266) board via the MQTT protocol.
4. The NodeMCU is connected with the user's device using the Blynk IoT platform. The Blynk client application is installed within the device which enables the user to monitor the water level as well as control the flow through the solenoid valve.

4.2.2 NodeMCU Implementation

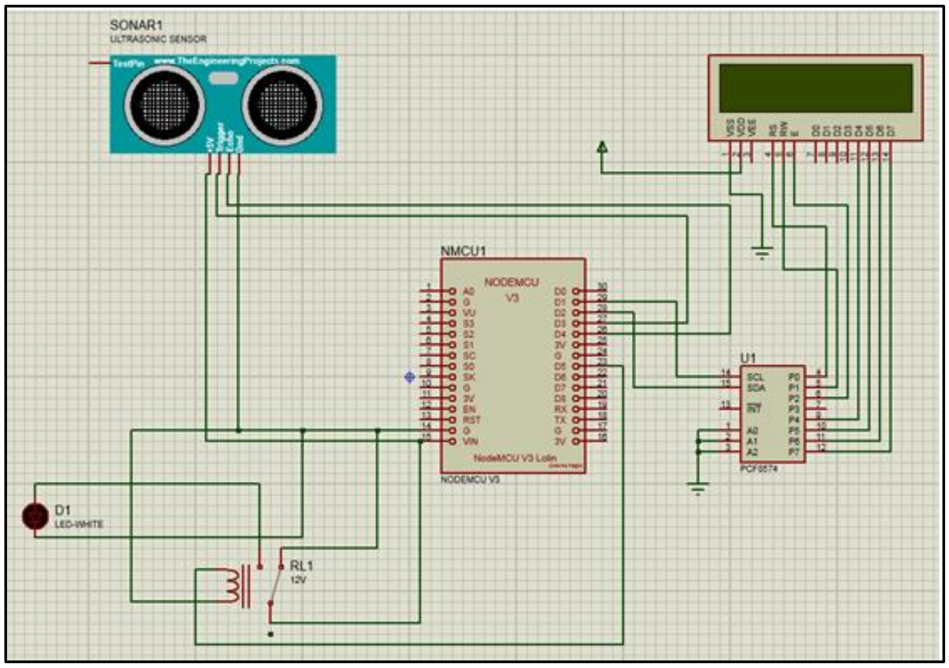


Figure 4.2. 2 Circuit Diagram of the System with NodeMCU

1. The Ultrasonic Sensor placed within the tank acts as the main sensing element of the system. The water level determined by the sensor is visible on the LCD screen via the NodeMCU.
2. The LED (simulating the solenoid valve) and the Relay connected to the NodeMCU are the Mechanical Assembly responsible to allow / disallow the water flow from the tank. The Relay coil, once energized, causes the switching action responsible for the LED to light up (simulating the closing of the solenoid valve).
3. The LCD display is connected to the NodeMCU using the I2C module. The water level is indicated upon the screen, thereby easing water wastage detection by monitoring the levels.
4. The NodeMCU acts as the central element which enables sensing the water level and controlling the valve in order to allow / disallow the flow.

5. COMPONENTS & TOOLS

Table 5.1 COMPONENTS

Sr. No.	Component	Usage
1.	Breadboard	It is used to connect the hardware components required for the implementation of the system.
2.	Connecting Wires	To connect the various components in the system.
3.	2- way Solenoid Valve	To control the flow of water in the pipe.
4.	Supporting Structure	To construct the base of the assembly (pipes, etc.)
5.	NodeMCU (ESP8266)	It is used to wirelessly connect the user’s device to the Mechanical Assembly and Sensor utilizing the Blynk app.
6.	12V Rechargeable Battery	It is used to power the Solenoid Valve used in the Mechanical Assembly.
7	Ultrasonic Sensor	It is used to detect the water level.
8	Jumper Wires	To connect the various components in the system.
9	2-channel Relay Module	It is used to control the working of the Solenoid Valves.

Table 5.2 ARCHITECTURE OF RASPBERRY PI

Raspberry Pi is a mini single-chip computer. It is developed by the Raspberry Pi Foundation in the United Kingdom along with the association of Broadcom. Raspberry Pi Zero is the smallest chipset in the Raspberry Pi series and is 40% faster than the original Raspberry Pi but nearly half of its size.

The Raspberry Pi Zero supports mini connectors (like mini HDMI, mini USB power, and USB on-the-go port) to save more space. And the 40pin GPIO is unpopulated which provides the flexibility to use only the connections that the project requires. It consists of a 1GHz BCM2835 single-core processor, 512 MB RAM, mini-HDMI, USB On-The-Go ports, and a camera connector.
One powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the extreme right edge of the board. Like every Raspberry Pi chipset, Zero consists of a 40-pin GPIO. The standard interface for connecting a single-board computer or microprocessor to other devices is through General-Purpose Input/Output (GPIO) pins. GPIO pins do not have a specific function and can be customized using the software.

Specifications of Raspberry Pi Zero:

- BCM2835 single-core processor :1GHz ARM11 core
- 512MB of LPDDR2 SDRAM
- A micro-SD card slot
- A mini-HDMI socket for 1080p60 video output
- Micro-USB sockets for data and power
- An unpopulated 40-pin GPIO header
- An unpopulated composite video header
- Dimension: 65mm x 30mm x 5mm
- Composite video and reset headers
- CSI camera connector (v1.3 only)

The Pi Zero is small and thin 65mm long x 30mm wide x 5mm thick, way smaller than the Pi 2 or B+ and even smaller than the A+, its 60% the size of the A+ and about 40% the size of the Pi 2 or B+. To keep the Pi Zero low cost, the processor and RAM are kept basic. Instead of the Pi 2's zippy quad core ARM v7, it has a single-core 1GHz ARM (same processor in the Pi Model B+ and A+). It also has 512 MB of RAM with a 'package-on-package' setup. the Pi Zero does not have a USB Hub built in which means you get one USB port.

Raspberry PI Zero Power Pins:

The board consists of two 5V pins, two 3V3 pins, and 9 ground pins (0V), which are unconfigurable.

5V: The 5v pins directly deliver the 5v supply coming from the mains adaptor. This pin can use to power up the Raspberry Pi zero, and it can also use to power up other 5v devices.

3.3V: The 3v pin is there to offer a stable 3.3v supply to power components and to test LEDs.

GND: Ground is commonly referred to as GND. GND pin is from where all voltages are measured and it also completes an electrical circuit.

Raspberry Pi Zero Input/Outputs pins:

A GPIO pin that is set as an input pin, receives the incoming voltage signal sent by the device connected to this pin. A voltage between 1.8V and 3.3V will be read by the Raspberry Pi as HIGH and if the voltage is lower than 1.8V will be read as LOW.

A GPIO pin set as an output pin sends the voltage signal as high (3.3V) or low (0V). When this pin is set to HIGH, the voltage at the output is 3.3V and when set to LOW, the output voltage is 0V.

Other Important Pins on Raspberry Pi Zero:

Along with the simple function of input and output pins, the GPIO pins can also perform a variety of alternative functions. Some specific pins are:

PWM (pulse-width modulation) Pins:

- Software PWM available on all pins
- Hardware PWM available on these pins: GPIO12, GPIO13, GPIO18, GPIO19

SPI PINS on R-Pi Zero:

SPI (Serial Peripheral Interface) is another protocol used for master-slave communication. It is used by the Raspberry pi board to quickly communicate between one or more peripheral devices. Data is synchronized using a clock (SCLK at GPIO11) from the master (RPi) and the data is sent from the Pi to our SPI device using the MOSI (Master Out Slave In) pin. If the SPI device needs to communicate back to Raspberry Pi, it sends the data back using the MISO (Master in Slave Out) pin.

- SPI0: GPIO9 (MISO), GPIO10 (MOSI), GPIO11 (SCLK), GPIO8 (CE0), GPIO7 (CE1)
- SPI1: GPIO19 (MISO), GPIO20 (MOSI), GPIO21 (SCLK), GPIO18 (CE0), GPIO17 (CE1), GPIO16 (CE2)

I2C Pins on R-Pi Zero:

I2C is used by the Raspberry Pi board to communicate with devices that are compatible with Inter-Integrated Circuit (a low-speed two-wire serial communication protocol). This communication standard requires master-slave roles between both the devices. I2C has two connections: SDA (Serial Data) and SCL (Serial Clock). They work by sending data to and using the SDA connection, and the speed of data transfer is controlled via the SCL pin.

- Data: (GPIO2), Clock (GPIO3)
- EEPROM Data: (GPIO0), EEPROM Clock (GPIO1)

UART Pins on R-Pi Zero:

Serial communication or the UART (Universal Asynchronous Receiver / Transmitter) pins provide a way to communicate between two microcontrollers or the computers. TX pin is used to transmit the serial data and RX pin is used to receive serial data coming from a different serial device.

- TX (GPIO14)
- RX (GPIO15)

Table 5.3 SOFTWARE	
Canva	To design the block diagrams (systems architecture).
Proteus	To virtually simulate the working of the entire system circuitry.
Arduino IDE	To upload the code in the NodeMCU (ESP8266)
Blynk App	To control the on and off operation of the valves and to display the water level.

6. IMPLEMENTATION STEPS

6.1 SOFTWARE SIMULATION (PROTEUS)

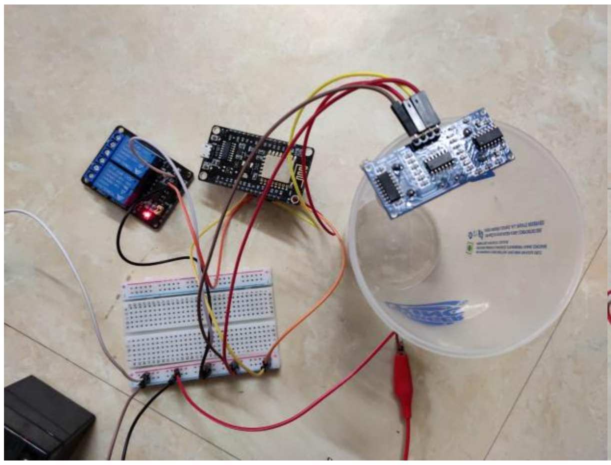
1. Download and install the Proteus 8 software into the device.
2. Download the NodeMCU libraries for Proteus available as zip files online. Extract the zip file into your device and add it in the Proteus file location and it will be available to use.
3. Click on “New Project” and select all the components required from the components list: NodeMCU, Ultrasonic Sensor, Relay, I2C LCD display and LED (for the simulation of the solenoid valve).
4. Drag the components one after the other in the schematic capture for making all the connections.
5. Connect the Vcc of the sensor to the Vcc terminal of the NodeMCU, the Trigger input (TRIG) of the sensor to the D3 port of the NodeMCU, the Echo terminal of the sensor to the D4 port of the NodeMCU and the Ground (GND) terminal of the sensor to the Gnd terminal of the NodeMCU.
6. For the 2 channel Relay Module, connect the Vcc and Gnd terminals of the relay module to the NodeMCU and then connect the IN1 terminal of the relay module to the D5 port of the NodeMCU. Connect the other two ends to the LED.
7. Next, for the I2C module LCD display make the Vcc and Gnd connections. Then connect the SDA to the D2 port of the NodeMCU and the SCL to the D1 port of the NodeMCU.
8. For the simulation, compile the project code on the Arduino IDE which will generate a temporary ‘.hex’ file into your device. Copy the file to the NodeMCU in the software.
9. Proceed by running the simulation on the software, to see the working of the system.

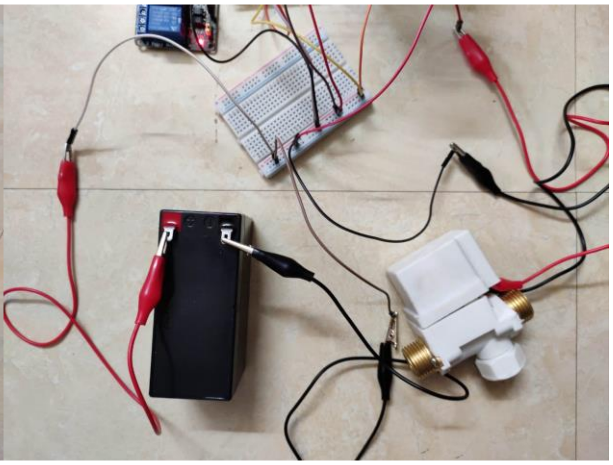
6.2 HARDWARE IMPLEMENTATION

1. Mount the NodeMCU (ESP8266) on a 400 tie-point breadboard.
2. Take two male to male jumper wires and connect the Vin and GND terminals of the NodeMCU on the breadboard.
3. Next take four male to female jumper wires and the ultrasonic sensor. Connect the Vcc of the sensor to the Vcc terminal on the breadboard, connect the Trig of the sensor to the D3 port of the NodeMCU, connect the Echo terminal of the sensor to the D4 port of the

NodeMCU and lastly the GND terminal of the sensor to the GND terminal on the breadboard.

4. Now, take the 2 channel Relay Module and 3 male to female jumper wires. Connect the Vcc and Gnd terminals of the relay module on the breadboard and then connect the IN1 terminal of the relay module to the D5 port of the NodeMCU.
5. The Normally Open terminal of the Relay module is connected to the Positive of the Solenoid Valve and the Negative terminal of the Solenoid Valve is connected to Ground.
6. Now using a USB Cable, connect the NodeMCU to the computer.
7. For the next step, to set up the Blynk Application, download the Blynk App from the play store or the app store. After installing the app follow the steps below -
   I. First, click the “New Project” button. After, enter the project name as you like. Then select the device and connection type. Finally, click the “Confirm” button.
   II. Next, click the “+” button and include a “Button” and “Level V” widget. OK, let’s set up this button. For that, click this “Button”. Now, enter the button name as you like. After, select “Virtual V0” for the PIN and drag the button below it to the “Switch” side.
   III. Now for setting up the “level V” widget. For that, click this widget. Now, enter the widget name as you like. Then, select “Virtual V1” for the PIN and enter your water tank level. Then turn ON the flip axis button.
   IV. Lastly, arrange these widgets as required.
8. Now, after setting up the Blynk application, open Arduino Ide and enter the project code. Enter the auth token of the Blynk Application which will be sent to you via email. Enter your WIFI Name and Password in the code.
9. Next, in the Tools section, select the board as NodeMCU1.0 (ESP-12E Module) and the port as COM2.
10. Finally upload this code to the NodeMCU Board and then run the Blynk app interface which we created earlier.


(a) Hardware Panel (NodeMCU, Relay Module and Ultrasonic Sensor


(b) Hardware Panel (Solenoid Valve and Power Supply

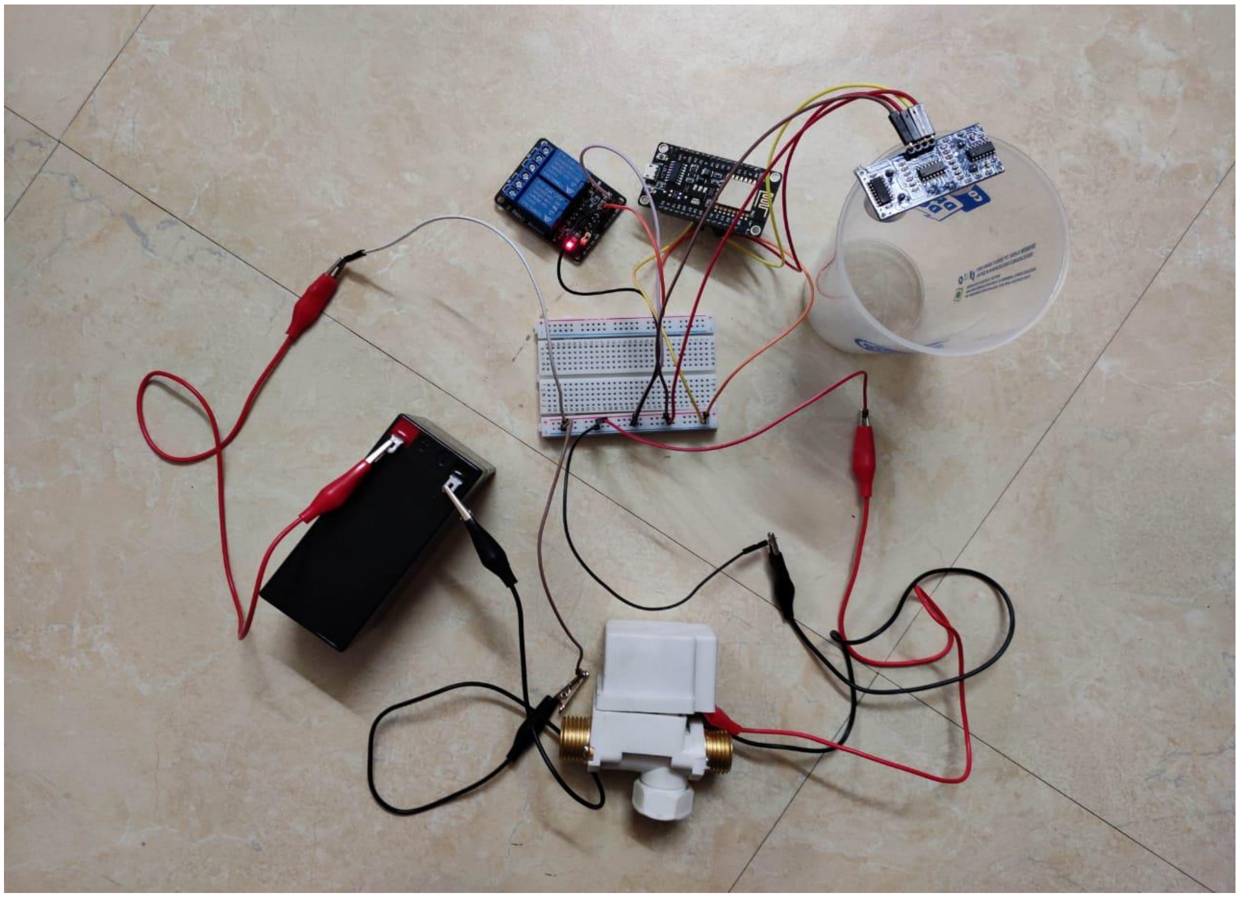

(b) Hardware Panel (Complete Assembly)

Figure 6.2. 1 Hardware Implementation

7. TROUBLESHOOTING & DEBUGGING

7.1 PROBLEMS ENCOUNTERED IN THE PROJECT

- Initially, we needed a **fast, wireless and reliable mechanism** to control the flow of water within the pipes. We had to ensure the least amount of water wastage and also required a way to **remotely control the flow of water** within the pipes.
- **Raspberry Pi** offers various versions including 3, 3B, 4, 4B, Zero, etc. so choosing an appropriate version for our system architecture was essential.
- A **Water Meter** works by measuring the speed of water flowing through the pipe that causes a piston or turbine to rotate. The main issue in this implementation included the **high cost and maintenance** due to which this implementation was discarded as well.
- We had to ensure that the system notifies **not only the user (flat owner), but also the concerned society personnel** so that the society personnel may shut off the water supply to the tank in any unforeseen situations.

7.2 STEPS TAKEN TO SOLVE THE PROBLEMS

- **Solenoid Valves** were finally decided to be used to regulate the flow of water within the pipes. It was chosen because of its **speed**, **efficiency**, **less maintenance**, **less cost** and the **ability for remote control**.
- A **NodeMCU ESP8266** is a low-cost open source IoT platform. The ESP12-E **NodeMCU** Kit is one of the most used **ESP8266** development boards. Due to its wireless abilities, it is used in place of Raspberry Pi Zero.
- 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. It is a feasible option to detect the water level.
- The **Blynk App allows** you to create amazing interfaces for your projects using various widgets they provide. It is used to display the water level on your smartphone and also to control the flow of water.
- **Proteus** is chosen for simulation purposes because it allows for **remote work and collaboration among the team members**. Also, an **intuitive UI** makes it easy to operate and work on.

8. CONCLUSION

- We have been successful in creating a centralized partially-wireless system to counter the water wastage issues faced in many Societies. The proposed retort consists of two implementations as a solution - Raspberry Pi Implementation and NodeMCU-based Implementation.
- The Ideation Phase for both the implementations have been completed, including Circuit Design and System Architecture Planning of the proposed solutions. Also, the Virtual Simulation for the complete system architecture (NodeMCU-based implementation) has been successfully completed.
- Finally, the Hardware Implementation of the NodeMCU-based system has been completed. The Blynk Client Application (installed within the user's device) is successfully able to monitor the water levels and control the flow of water.

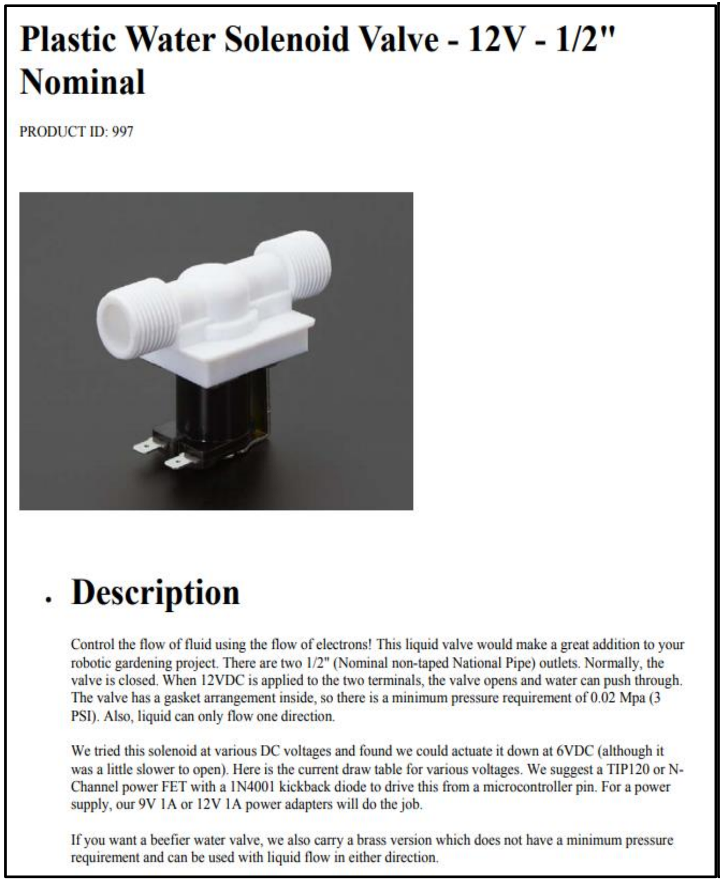
9. REFERENCES

1. Circuits Today. Water Level Controller using Arduino. (2014). Accessed: Feb. 21, 2021. [Online].Available: https://www.circuitstoday.com/water-level-controller-using-arduino#:~:text=This%20article%20is%20a%20about,when%20the%20tank%20is%20full
2. Arduino.cc/Project Hub. Automatic Water Level using Arduino.(2018). Accessed: Feb. 21,2021.[Online].Available: https://create.arduino.cc/projecthub/karthickcj0083/automatic-water-level-controller-b2508d
3. GitHub. Blynk Water Level Controller. (Aug 21,2018). Accessed: Mar. 01,2021. [Online]. Available: https://github.com/pranz24/Blynk-Water-Level-Sensor
4. Instructables. IOT based Water Level Controller Using NodeMCU (Apr. 27, 2019). Accessed: Mar. 01, 202. [Online]. Available: https://www.instructables.com/IOT-Based-Water-Level-Controller-Using-NodeMCU-ES
5. Home Automation. Smart Water Tank with Real-Time Dashboard. (Jun. 30, 2020). Accessed: Apr. 14, 2021. [Online Video]. Available: https://youtu.be/7uLuwq3Zd_M
6. Tinker and Build. Arduino Solenoid Valve Circuit: How to control water flow with an Arduino. (Nov. 22, 2017). Accessed: Apr. 14, 2021. [Online Video]. Available: https://youtu.be/ioSYlxHlYdI
7. Tinker and Build. How to use an Arduino Relay Module. (Feb 05,2018). Accessed: Apr. 14, 2021 [Online Video]. Available: https://www.youtube.com/watch?v=0BNcI8jMcXE&t=0s
8. How to Electronics. IoT Water Flow Meter using NodeMCU ESP8266 & Water Flow Sensor.(Feb 05,2018). Accessed: Apr. 14, 2021 [Online Video] Available: https://youtu.be/SBWgySTPhNM
9. Robotic Nation. Arduino to Raspberry Pi Serial Communication .(Apr 10,2020). Accessed: Apr. 14, 2021 [Online Video] Available: https://youtu.be/k6t9hNteEX0
10. SriTu Tech. Water tank level monitoring system with NodeMCU and Blynk application. (Dec 20,2020). Accessed: Apr. 14, 2021 [Online Video] Available: https://youtu.be/Iuxl0k4lnqA

11. Homemade Electronics. Download and Install Proteus 8.11 Latest Version full Software with Arduino Libraries for Free. (Dec 20,2020). Accessed: Apr. 20, 2021 [Online Video] Available: https://youtu.be/A2KrMkxZQmw
12. Talking Stuff. Ultrasonic Water Level Indicator using HC-SR04 & Arduino.(Jul 28,2020). Accessed: Apr. 30, 2021 [Online Video] Available: https://youtu.be/o5Qo_I0TdOY
13. Arduino Experiments. Arduino Automatic Water Pump System. (Dec 20,2020). Accessed: Apr. 30, 2021 [Online Video] Available: https://youtu.be/FkPbwRjNjCE
14. VTip.Automatic watering controller using Arduino - Uno Board. (Sep 11,2019). Accessed: Apr. 30, 2021 [Online Video] Available: https://youtu.be/EvBiTLfXPOo
15. Last Minute Engineers. Insight Into ESP8266 NodeMCU Features & Using It With Arduino IDE. Accessed: May. 01,2021. [Online]. Available:https://lastminuteengineers.com/esp8266-nodemcu-arduino-tutorial/
16. A Water Level Detection: IoT Platform Based on Wireless Sensor Network, IEEE Explore, [Online]. Available: https://ieeexplore.ieee.org/document/8878559
17. IoT Based flow control System, IEEE Explore , [Online]. Available: https://ieeexplore.ieee.org/document/8389671

10. APPENDIX

2-way Solenoid Valve (12V)



Plastic Water Solenoid Valve - 12V - 1/2" Nominal

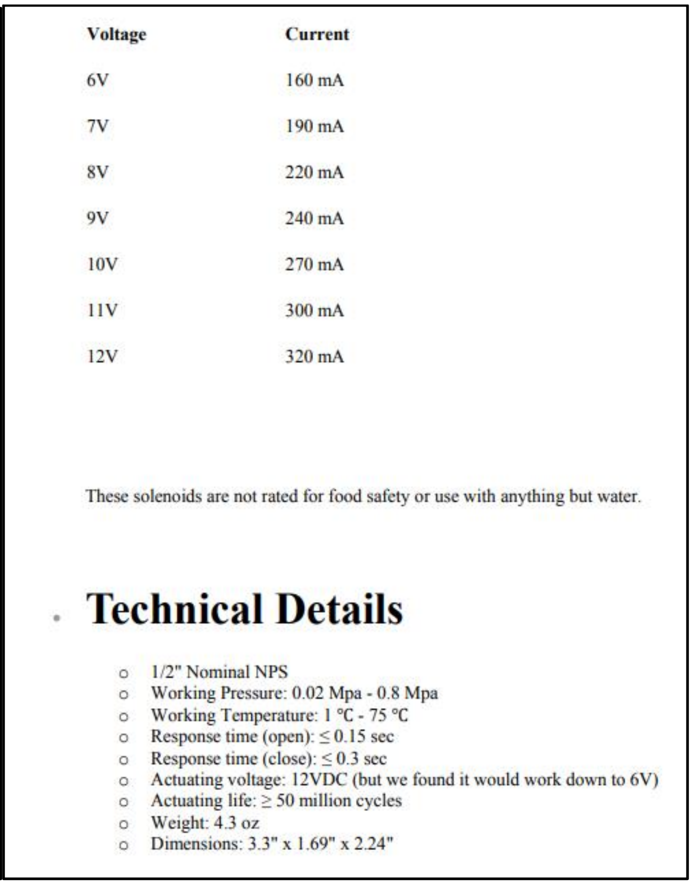
PRODUCT ID: 997

- **Description**

Control the flow of fluid using the flow of electrons! This liquid valve would make a great addition to your robotic gardening project. There are two 1/2" (Nominal non-taped National Pipe) outlets. Normally, the valve is closed. When 12VDC is applied to the two terminals, the valve opens and water can push through. The valve has a gasket arrangement inside, so there is a minimum pressure requirement of 0.02 Mpa (3 PSI). Also, liquid can only flow one direction.

We tried this solenoid at various DC voltages and found we could actuate it down at 6VDC (although it was a little slower to open). Here is the current draw table for various voltages. We suggest a TIP120 or N-Channel power FET with a 1N4001 kickback diode to drive this from a microcontroller pin. For a power supply, our 9V 1A or 12V 1A power adapters will do the job.

If you want a beefier water valve, we also carry a brass version which does not have a minimum pressure requirement and can be used with liquid flow in either direction.

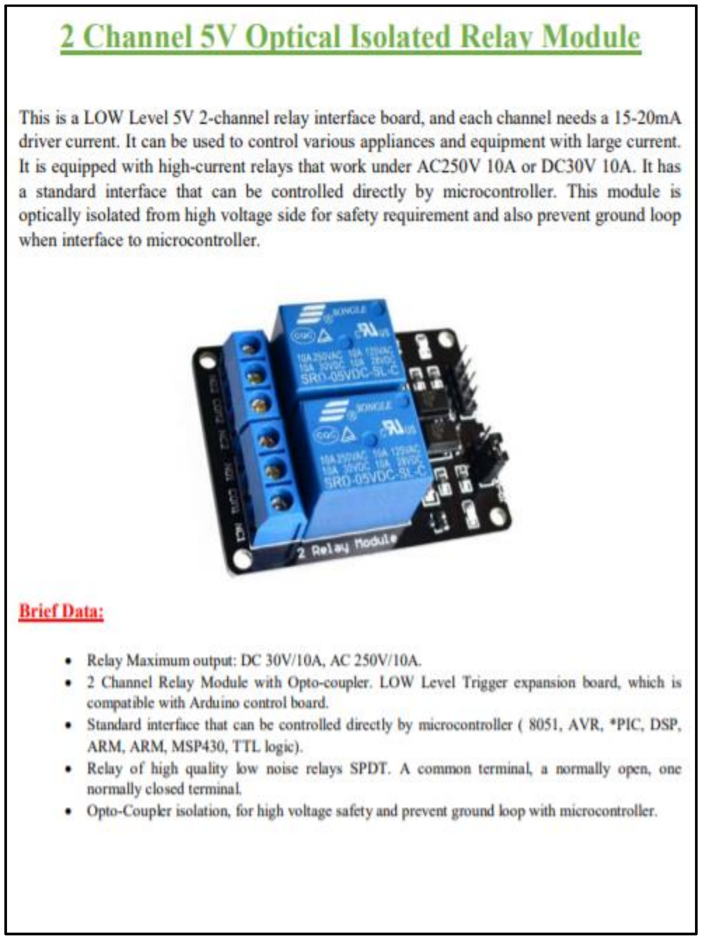


Voltage	Current
6V	160 mA
7V	190 mA
8V	220 mA
9V	240 mA
10V	270 mA
11V	300 mA
12V	320 mA

These solenoids are not rated for food safety or use with anything but water.

- **Technical Details**
  - 1/2" Nominal NPS
  - Working Pressure: 0.02 Mpa - 0.8 Mpa
  - Working Temperature: 1 ℃ - 75 ℃
  - Response time (open): ≤ 0.15 sec
  - Response time (close): ≤ 0.3 sec
  - Actuating voltage: 12VDC (but we found it would work down to 6V)
  - Actuating life: ≥ 50 million cycles
  - Weight: 4.3 oz
  - Dimensions: 3.3" x 1.69" x 2.24"

2-channel SPDT Relay Module

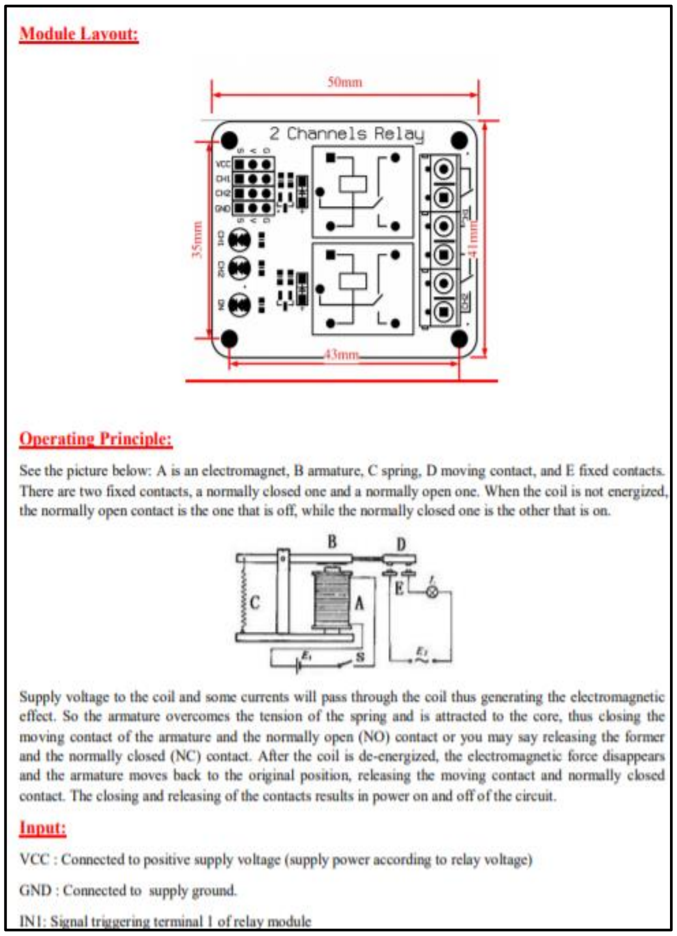


2 Channel 5V Optical Isolated Relay Module

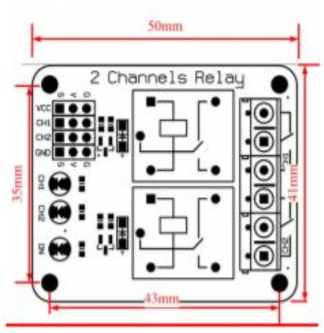
This is a LOW Level 5V 2-channel relay interface board, and each channel needs a 15-20mA driver current. It can be used to control various appliances and equipment with large current. It is equipped with high-current relays that work under AC250V 10A or DC30V 10A. It has a standard interface that can be controlled directly by microcontroller. This module is optically isolated from high voltage side for safety requirement and also prevent ground loop when interface to microcontroller.

Brief Data:

- Relay Maximum output: DC 30V/10A, AC 250V/10A.
- 2 Channel Relay Module with Opto-coupler. LOW Level Trigger expansion board, which is compatible with Arduino control board.
- Standard interface that can be controlled directly by microcontroller (8051, AVR, *PIC, DSP, ARM, ARM, MSP430, TTL logic).
- Relay of high quality low noise relays SPDT. A common terminal, a normally open, one normally closed terminal.
- Opto-Coupler isolation, for high voltage safety and prevent ground loop with microcontroller.

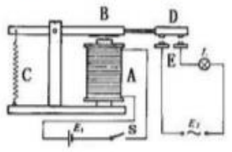


Module Layout:



Operating Principle:

See the picture below: A is an electromagnet, B armature, C spring, D moving contact, and E fixed contacts. There are two fixed contacts, a normally closed one and a normally open one. When the coil is not energized, the normally open contact is the one that is off, while the normally closed one is the other that is on.



Supply voltage to the coil and some currents will pass through the coil thus generating the electromagnetic effect. So the armature overcomes the tension of the spring and is attracted to the core, thus closing the moving contact of the armature and the normally open (NO) contact or you may say releasing the former and the normally closed (NC) contact. After the coil is de-energized, the electromagnetic force disappears and the armature moves back to the original position, releasing the moving contact and normally closed contact. The closing and releasing of the contacts results in power on and off of the circuit.

Input:

VCC : Connected to positive supply voltage (supply power according to relay voltage)

GND : Connected to supply ground.

IN1: Signal triggering terminal 1 of relay module

IN2: Signal triggering terminal 2 of relay module

Output:

Each module of the relay has one NC (normally close), one NO (normally open) and one COM (Common) terminal. So there are 2 NC, 2 NO and 2 COM of the channel relay in total. NC stands for the normal close port contact and the state without power. NO stands for the normal open port contact and the state with power. COM means the common port. You can choose NC port or NO port according to whether power or not.

Testing Setup:

When a low level is supplied to signal terminal of the 2-channel relay, the LED at the output terminal will light up. Otherwise, it will turn off. If a periodic high and low level is supplied to the signal terminal, you can see the LED will cycle between on and off.

For Arduino：

Step 1：

Connect the signal terminal IN1、IN2 of 2-channel relay to digital pin 4 & 5 of the Arduino Uno or ATMega2560 board, and connect an LED at the output terminal.

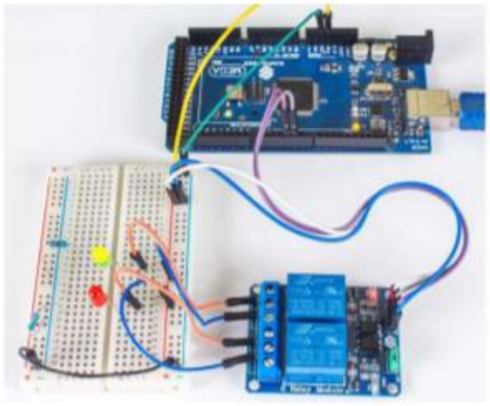
IN1> 4

IN2> 5

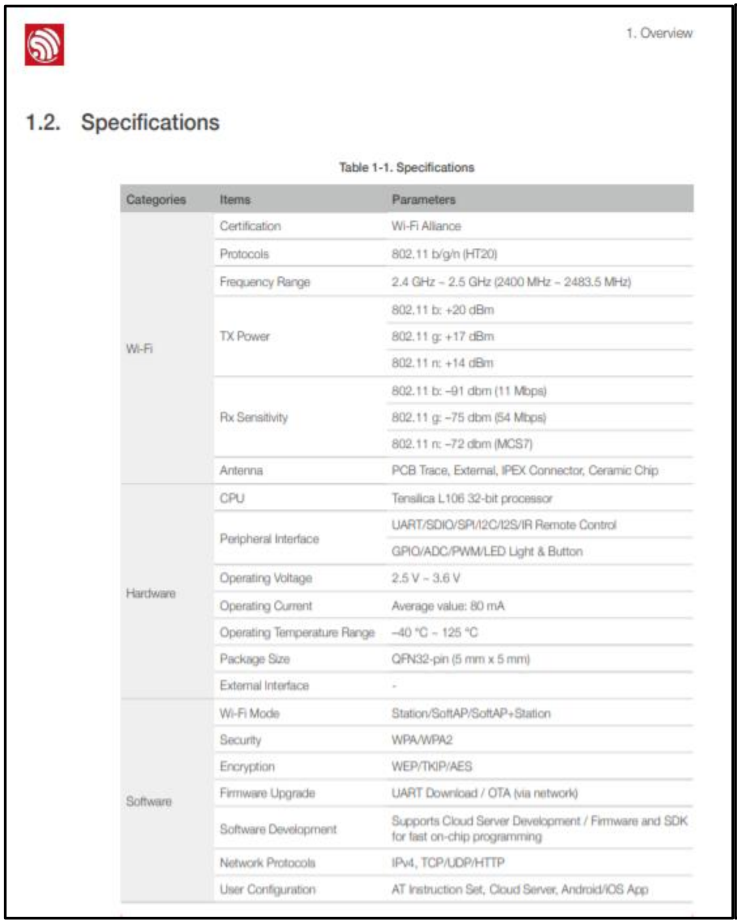
Step 2:

Upload the sketch "text_code" to the Arduino Uno or ATMega2560 board.Then you can see the LED cycle between on and off.

The actual figure is shown below:



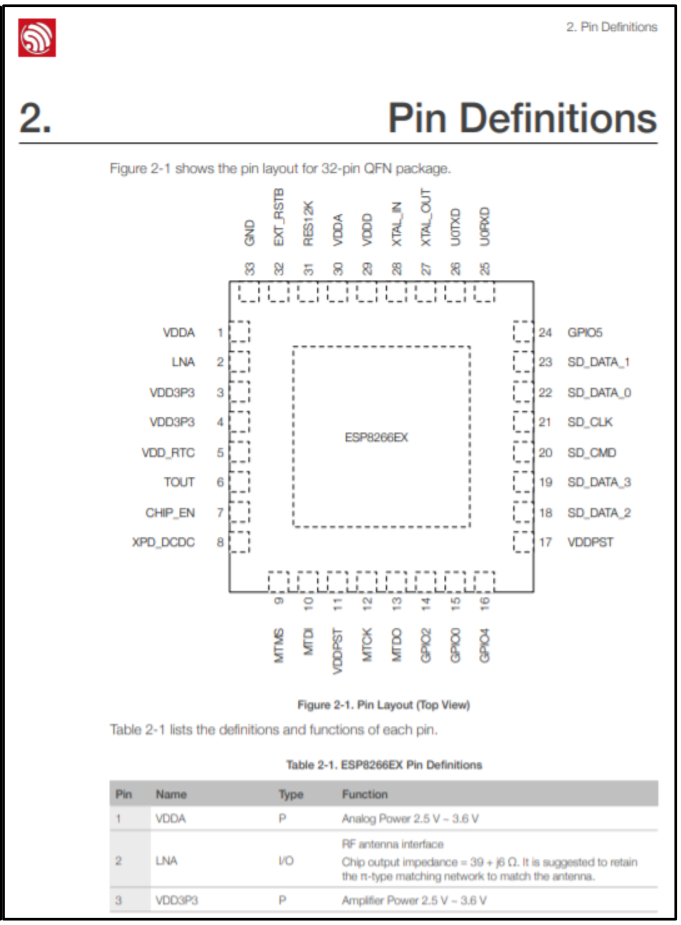
NodeMCU ESP8266



1.2. Specifications

Table 1-1. Specifications

Categories	Items	Parameters
Wi-Fi	Certification	Wi-Fi Alliance
	Protocols	802.11 b/g/n (HT20)
	Frequency Range	2.4 GHz ~ 2.5 GHz (2400 MHz ~ 2483.5 MHz)
	TX Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: –91 dbm (11 Mbps)
		802.11 g: –75 dbm (54 Mbps)
		802.11 n: –72 dbm (MCS7)
	Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip
Hardware	CPU	Tensilica L106 32-bit processor
	Peripheral Interface	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/ADC/PWM/LED Light & Button
	Operating Voltage	2.5 V ~ 3.6 V
	Operating Current	Average value: 80 mA
	Operating Temperature Range	–40 °C ~ 125 °C
	Package Size	QFN32-pin (5 mm x 5 mm)
	External Interface	-
Software	Wi-Fi Mode	Station/SoftAP/SoftAP+Station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Software Development	Supports Cloud Server Development / Firmware and SDK for fast on-chip programming
	Network Protocols	IPv4, TCP/UDP/HTTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App



2. Pin Definitions

Figure 2-1 shows the pin layout for 32-pin QFN package.

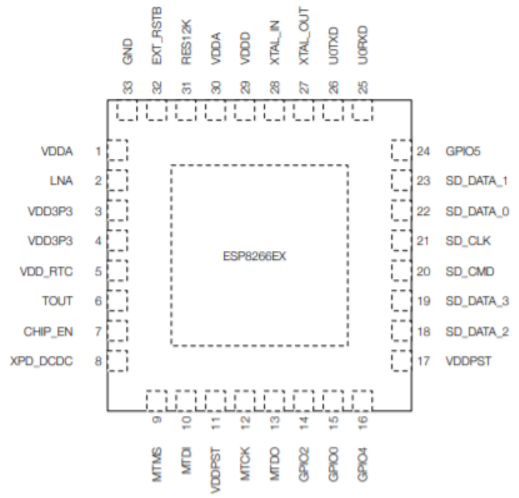
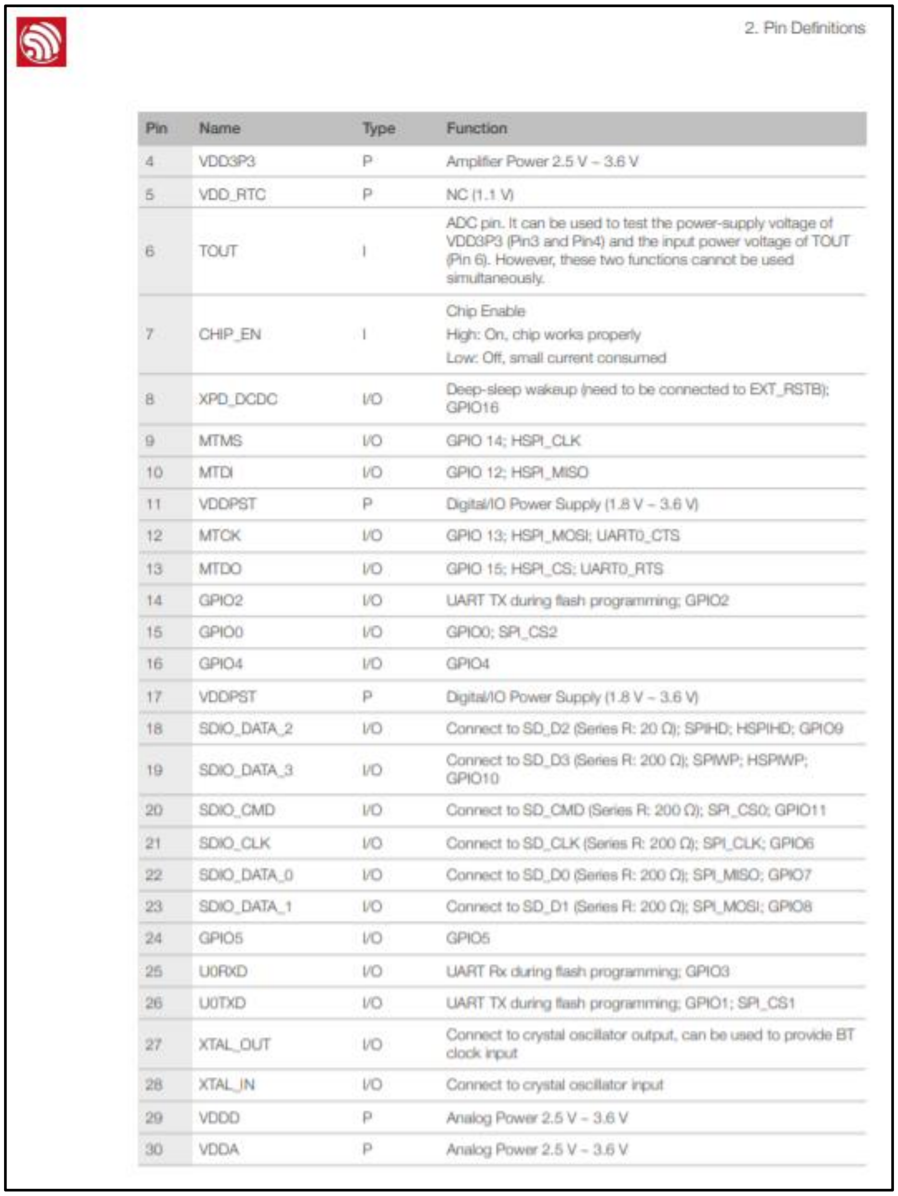


Figure 2-1. Pin Layout (Top View)

Table 2-1 lists the definitions and functions of each pin.

Table 2-1. ESP8266EX Pin Definitions

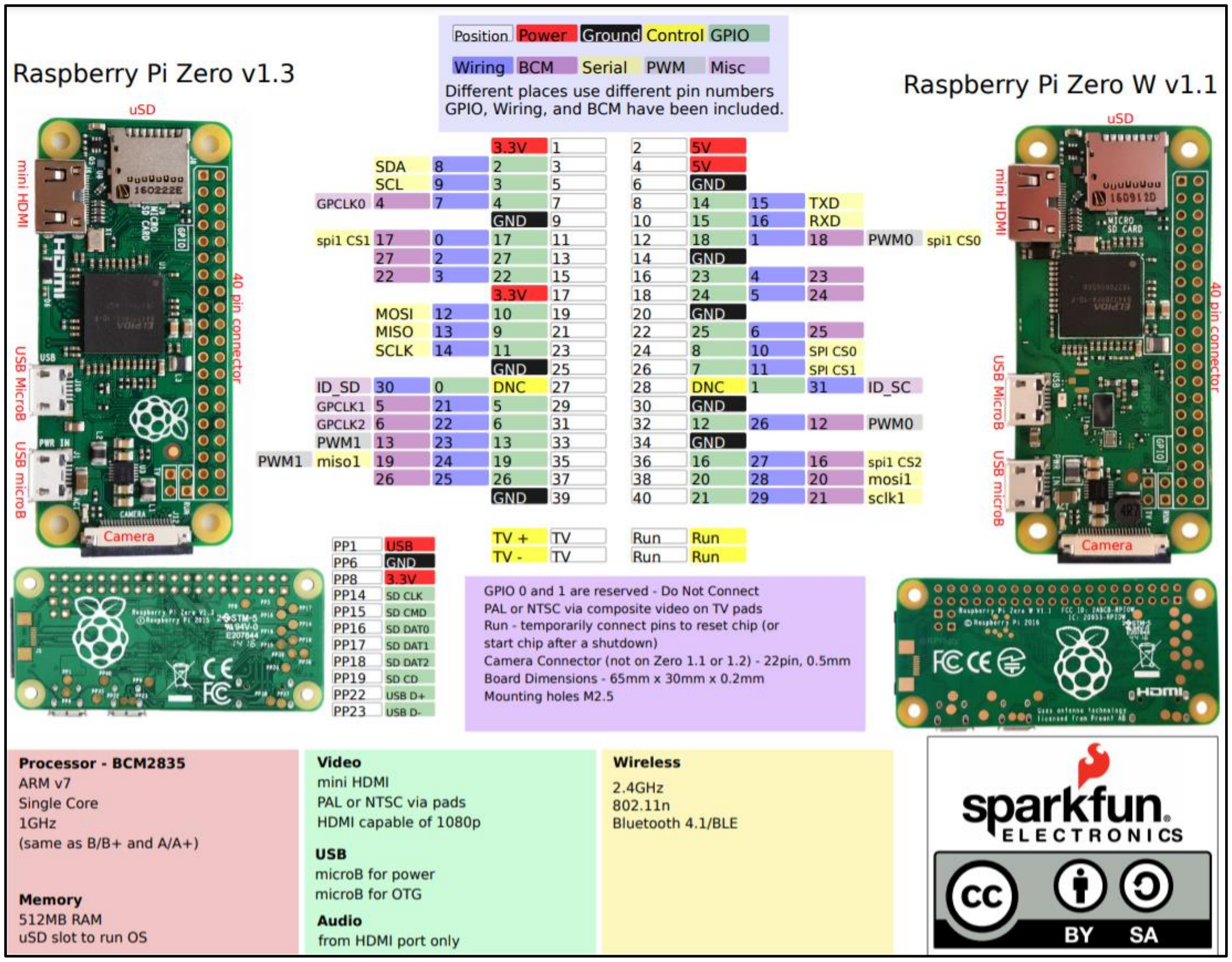
Pin	Name	Type	Function
1	VDDA	P	Analog Power 2.5 V ~ 3.6 V
2	LNA	I/O	RF antenna interface Chip output impedance = 39 + j6 Ω. It is suggested to retain the π-type matching network to match the antenna.
3	VDD3P3	P	Amplifier Power 2.5 V ~ 3.6 V



2. Pin Definitions

Pin	Name	Type	Function
4	VDD3P3	P	Amplifier Power 2.5 V ~ 3.6 V
5	VDD_RTC	P	NC (1.1 V)
6	TOUT	I	ADC pin. It can be used to test the power-supply voltage of VDD3P3 (Pin3 and Pin4) and the input power voltage of TOUT (Pin 6). However, these two functions cannot be used simultaneously.
7	CHIP_EN	I	Chip Enable High: On, chip works properly Low: Off, small current consumed
8	XPD_DCDC	I/O	Deep-sleep wakeup (need to be connected to EXT_RSTB); GPIO16
9	MTMS	I/O	GPIO 14; HSPI_CLK
10	MTDI	I/O	GPIO 12; HSPI_MISO
11	VDDPST	P	Digital/IO Power Supply (1.8 V ~ 3.6 V)
12	MTCK	I/O	GPIO 13; HSPI_MOSI; UART0_CTS
13	MTDO	I/O	GPIO 15; HSPI_CS; UART0_RTS
14	GPIO2	I/O	UART TX during flash programming; GPIO2
15	GPIO0	I/O	GPIO0; SPI_CS2
16	GPIO4	I/O	GPIO4
17	VDDPST	P	Digital/IO Power Supply (1.8 V ~ 3.6 V)
18	SDIO_DATA_2	I/O	Connect to SD_D2 (Series R: 20 Ω); SPIHD; HSPIHD; GPIO9
19	SDIO_DATA_3	I/O	Connect to SD_D3 (Series R: 200 Ω); SPIWP; HSPIWP; GPIO10
20	SDIO_CMD	I/O	Connect to SD_CMD (Series R: 200 Ω); SPI_CS0; GPIO11
21	SDIO_CLK	I/O	Connect to SD_CLK (Series R: 200 Ω); SPI_CLK; GPIO6
22	SDIO_DATA_0	I/O	Connect to SD_D0 (Series R: 200 Ω); SPI_MISO; GPIO7
23	SDIO_DATA_1	I/O	Connect to SD_D1 (Series R: 200 Ω); SPI_MOSI; GPIO8
24	GPIO5	I/O	GPIO5
25	U0RXD	I/O	UART Rx during flash programming; GPIO3
26	U0TXD	I/O	UART TX during flash programming; GPIO1; SPI_CS1
27	XTAL_OUT	I/O	Connect to crystal oscillator output, can be used to provide BT clock input
28	XTAL_IN	I/O	Connect to crystal oscillator input
29	VDDD	P	Analog Power 2.5 V ~ 3.6 V
30	VDDA	P	Analog Power 2.5 V ~ 3.6 V

Raspberry Pi Zero



Position	Power	Ground	Control	GPIO	Wiring	BCM	Serial	PWM	Misc

Different places use different pin numbers GPIO, Wiring, and BCM have been included.

				3.3V	1	2	5V				
	SDA	8		2	3	4	5V				
	SCL	9		3	5	6	GND				
	GPCLK0	4	7	4	7	8	14	15	TXD		
				GND	9	10	15	16	RXD		
	spi1 CS1	17	0	17	11	12	18	1	18	PWM0	spi1 CS0
		27	2	27	13	14	GND				
		22	3	22	15	16	23	4	23		
				3.3V	17	18	24	5	24		
	MOSI		12	10	19	20	GND				
	MISO		13	9	21	22	25	6	25		
	SCLK		14	11	23	24	8	10	SPI CS0		
				GND	25	26	7	11	SPI CS1		
	ID_SD	30	0	DNC	27	28	DNC	1	31	ID_SC	
	GPCLK1	5	21	5	29	30	GND				
	GPCLK2	6	22	6	31	32	12	26	12	PWM0	
	PWM1	13	23	13	33	34	GND				
PWM1	miso1	19	24	19	35	36	16	27	16	spi1 CS2	
		26	25	26	37	38	20	28	20	mosi1	
				GND	39	40	21	29	21	sclk1	
				TV +	TV	Run	Run				
				TV -	TV	Run	Run				

PP1	USB
PP6	GND
PP8	3.3V
PP14	SD CLK
PP15	SD CMD
PP16	SD DAT0
PP17	SD DAT1
PP18	SD DAT2
PP19	SD CD
PP22	USB D+
PP23	USB D-

GPIO 0 and 1 are reserved - Do Not Connect
PAL or NTSC via composite video on TV pads
Run - temporarily connect pins to reset chip (or start chip after a shutdown)
Camera Connector (not on Zero 1.1 or 1.2) - 22pin, 0.5mm
Board Dimensions - 65mm x 30mm x 0.2mm
Mounting holes M2.5

Processor - BCM2835
ARM v7
Single Core
1GHz
(same as B/B+ and A/A+)

Memory
512MB RAM
uSD slot to run OS

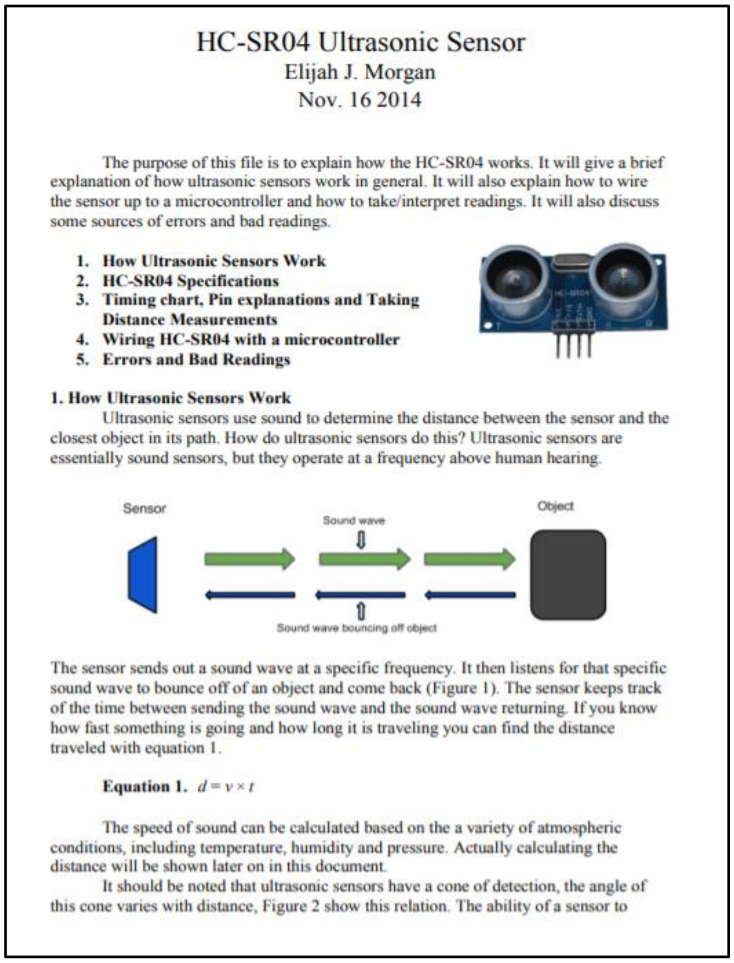
Video
mini HDMI
PAL or NTSC via pads
HDMI capable of 1080p

USB
microB for power
microB for OTG

Audio
from HDMI port only

Wireless
2.4GHz
802.11n
Bluetooth 4.1/BLE

Ultrasonic Sensor HC-SR04



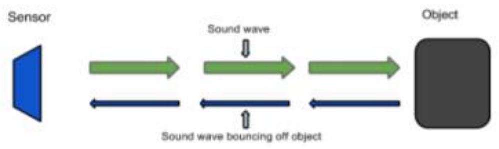
HC-SR04 Ultrasonic Sensor
Elijah J. Morgan
Nov. 16 2014

The purpose of this file is to explain how the HC-SR04 works. It will give a brief explanation of how ultrasonic sensors work in general. It will also explain how to wire the sensor up to a microcontroller and how to take/interpret readings. It will also discuss some sources of errors and bad readings.

1. **How Ultrasonic Sensors Work**
2. **HC-SR04 Specifications**
3. **Timing chart, Pin explanations and Taking Distance Measurements**
4. **Wiring HC-SR04 with a microcontroller**
5. **Errors and Bad Readings**

1. How Ultrasonic Sensors Work

Ultrasonic sensors use sound to determine the distance between the sensor and the closest object in its path. How do ultrasonic sensors do this? Ultrasonic sensors are essentially sound sensors, but they operate at a frequency above human hearing.

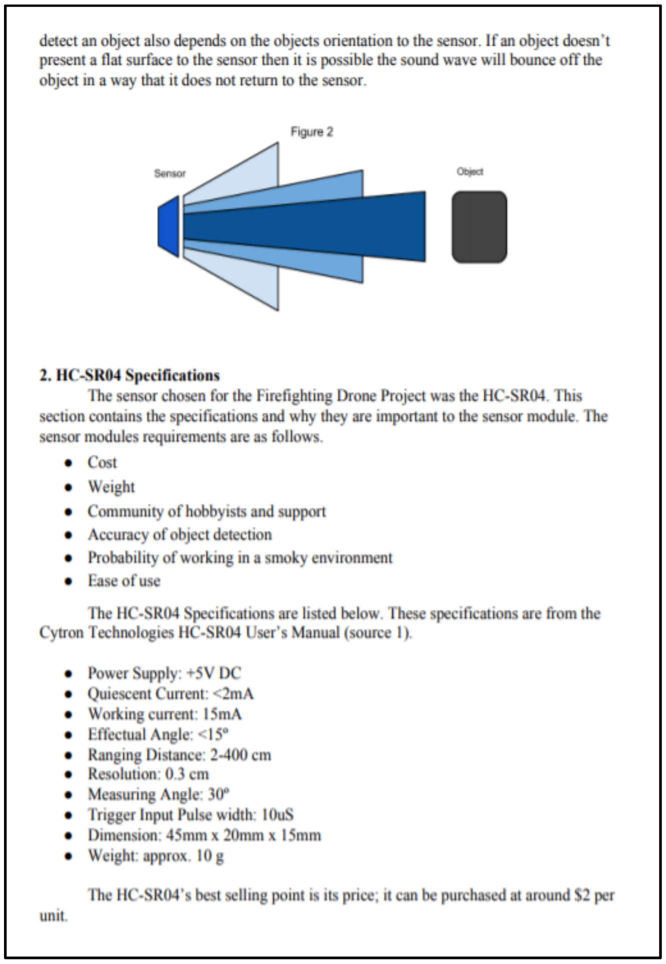


The sensor sends out a sound wave at a specific frequency. It then listens for that specific sound wave to bounce off of an object and come back (Figure 1). The sensor keeps track of the time between sending the sound wave and the sound wave returning. If you know how fast something is going and how long it is traveling you can find the distance traveled with equation 1.

Equation 1. $d = v \times t$

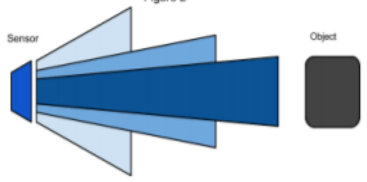
The speed of sound can be calculated based on the a variety of atmospheric conditions, including temperature, humidity and pressure. Actually calculating the distance will be shown later on in this document.

It should be noted that ultrasonic sensors have a cone of detection, the angle of this cone varies with distance, Figure 2 show this relation. The ability of a sensor to



detect an object also depends on the objects orientation to the sensor. If an object doesn't present a flat surface to the sensor then it is possible the sound wave will bounce off the object in a way that it does not return to the sensor.

Figure 2



2. HC-SR04 Specifications

The sensor chosen for the Firefighting Drone Project was the HC-SR04. This section contains the specifications and why they are important to the sensor module. The sensor modules requirements are as follows.

- Cost
- Weight
- Community of hobbyists and support
- Accuracy of object detection
- Probability of working in a smoky environment
- Ease of use

The HC-SR04 Specifications are listed below. These specifications are from the Cytron Technologies HC-SR04 User's Manual (source 1).

- Power Supply: +5V DC
- Quiescent Current: <2mA
- Working current: 15mA
- Effectual Angle: <15°
- Ranging Distance: 2-400 cm
- Resolution: 0.3 cm
- Measuring Angle: 30°
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm
- Weight: approx. 10 g

The HC-SR04's best selling point is its price; it can be purchased at around $2 per unit.

3. Timing Chart and Pin Explanations

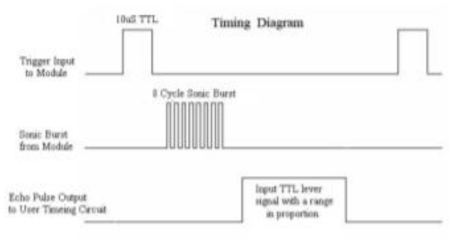
The HC-SR04 has four pins, VCC, GND, TRIG and ECHO; these pins all have different functions. The VCC and GND pins are the simplest -- they power the HC-SR04. These pins need to be attached to a +5 volt source and ground respectively. There is a single control pin: the TRIG pin. The TRIG pin is responsible for sending the ultrasonic burst. This pin should be set to HIGH for 10 μs, at which point the HC-SR04 will send out an eight cycle sonic burst at 40 kHZ. After a sonic burst has been sent the ECHO pin will go HIGH. The ECHO pin is the data pin -- it is used in taking distance measurements. After an ultrasonic burst is sent the pin will go HIGH, it will stay high until an ultrasonic burst is detected back, at which point it will go LOW.

Taking Distance Measurements

The HC-SR04 can be triggered to send out an ultrasonic burst by setting the TRIG pin to HIGH. Once the burst is sent the ECHO pin will automatically go HIGH. This pin will remain HIGH until the the burst hits the sensor again. You can calculate the distance to the object by keeping track of how long the ECHO pin stays HIGH. The time ECHO stays HIGH is the time the burst spent traveling. Using this measurement in equation 1 along with the speed of sound will yield the distance travelled. A summary of this is listed below, along with a visual representation in Figure 2.

1. Set TRIG to HIGH
2. Set a timer when ECHO goes to HIGH
3. Keep the timer running until ECHO goes to LOW
4. Save that time
5. Use equation 1 to determine the distance travelled

Figure 3
Source 2


10uS TTL
Timing Diagram
Trigger Input to Module
8 Cycle Sonic Burst
Sonic Burst from Module
Echo Pulse Output to User Timing Circuit
Input TTL lever signal with a range in proportion

Source 2

To interpret the time reading into a distance you need to change equation 1. The clock on the device you are using will probably count in microseconds or smaller. To use equation 1 the speed of sound needs to determined,which is 343 meters per second at standard temperature and pressure. To convert this into more useful form use equation 2 to change from meters per second to microseconds per centimeter. Then equation 3 can be used to easily compute the distance in centimeters.

Equation 2. $Distance = \frac{Speed}{170.15\,m} \times \frac{Meters}{100\,cm} \times \frac{1e6\,\mu S}{170.15\,m} \times \frac{58.772\,\mu S}{cm}$

Equation 3. $Distance = \frac{time}{58} = \frac{\mu s}{\mu s/cm} = cm$

4. Wiring the HC-SR04 to a Microcontroller

This section only covers the hardware side. For information on how to integrate the software side, look at one of the links below or look into the specific microcontroller you are using.

The HC-SR04 has 4 pins: VCC, GND, TRIG and ECHO.

1. VCC is a 5v power supply. This should come from the microcontroller
2. GND is a ground pin. Attach to ground on the microcontroller.
3. TRIG should be attached to a GPIO pin that can be set to HIGH
4. ECHO is a little more difficult. The HC-SR04 outputs 5v, which could destroy many microcontroller GPIO pins (the maximum allowed voltage varies). In order to step down the voltage use a single resistor or a voltage divider circuit. Once again this depends on the specific microcontroller you are using, you will need to find out its GPIO maximum voltage and make sure you are below that.

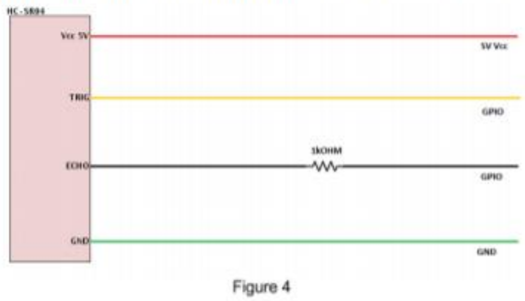


Figure 4