University of Southern Indiana
Pott College of Science, Engineering, and Education
Engineering Department
8600 University Boulevard
Evansville, Indiana 47712

Smart Hydroponic System

Gavin Vaal & Tyler Litkenhus

ENGR 491 – Senior Design

Spring 2024

Acknowledgments

We would like to thank the following for assisting with our project.

•Dr. Integlia

•Dr. Diersing

•Jamie Curry

•Meg Wagner

•Dr. Kuban

•Minka House staff

•USI’s Department of Engineering

Abstract

One of the biggest problems with gardening in its current state is its lack of accessibility towards the handicapped and elderly. How this problem was attacked was by taking a newer form of crop growing known as hydroponics and integrating it with smart technology. While designing the system, ways to make aspects of hydroponics accessible, such as the watering system and growing channels, to the handicapped and elderly were thought of. These aspects included the type of crop that grows in the system with microgreens being chosen for their quick grow time and ease of growth. Multiple sensors that could monitor the pump, flow rate, water level height, and PH levels of the system were implemented to allow less of a strain on the user. Lastly the physical design of the growing bed would allow almost anyone to run the system. With all these aspects considered, it was found that the smart hydroponic system could grow crops while being accessible to the handicapped and elderly.

Smart Hydroponic System

1.) INTRODUCTION

1.1: Overview

In the face of an aging global population, the need for solutions to address nutritional deficiencies in elderly and assisted living communities has become increasingly relevant. One avenue for meeting this challenge is the integration of hydroponic systems into these environments. This system would first be integrated into the Minka House, due to its current smart technology uses. The Minka house is a smart home showcase for living-in-place with integrated technology at USI. With its purpose being "for southwest Indiana to explore and to experiment with adaptable living and learning ideas and innovations in aging and wellness." To adapt this purpose, the system was designed to not only integrate well with their current technologies, but to also streamline the process of hydroponic growing and utilizing it to grow an easier crop, microgreens. Microgreens were used for this system because of the ease of planting and harvesting, very short yield time, and their nutrient density. To make this system smart and easy to manage, multiple sensors were used to monitor the watering system and a Bluetooth microcontroller to communicate with an app communicating these sensor readings to users.

1.2: Hydroponics

Hydroponics is a method of growing plants without soil using a nutrient-rich water solution to deliver nutrients directly to the plant roots. A base hydroponic system of this style usually consists of a water reservoir, a piping system, a growth medium and a growing bed. The water/nutrient solution is circulated through the system using a small pump. This hydroponics style requires a constant water stream, to supply nutrients to the crop and provides a growing medium got the roots of the plants. Hydroponics has become popular due to faster growth rates, higher crop yields, and its ability to grow in areas where the soil is not suitable for crop growth. Lastly a hydroponic system can be tailored to a specific plant through controlling PH levels and water usage to create the most suitable environment for the plant. Figure 1 shows a visual diagram of what goes into a hydroponic system

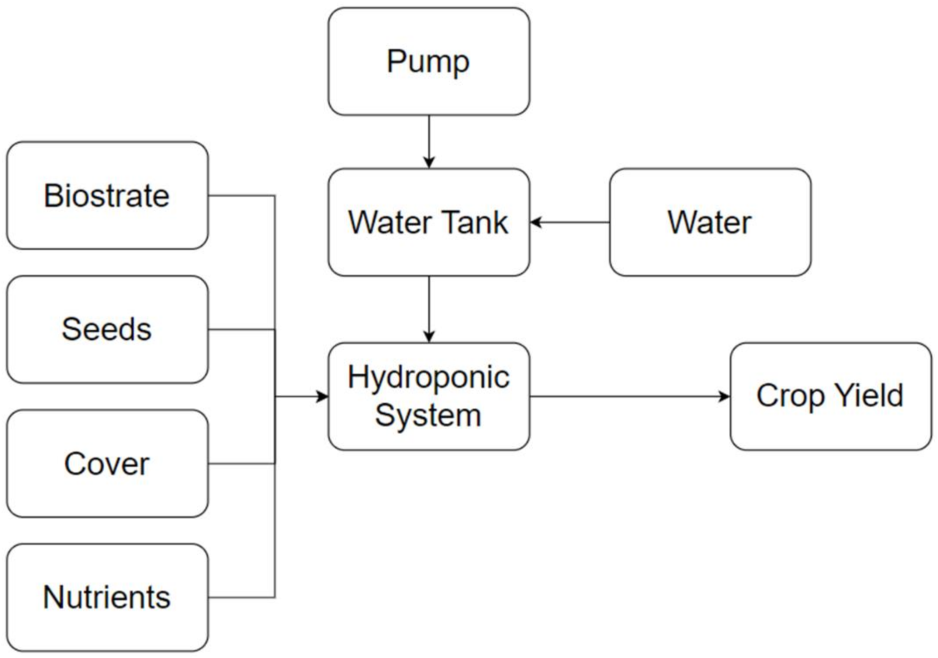


Figure 1: What Makes a Hydroponic System

1.3 Main Objective

The main objective was to design and create a functional and accessible smart hydroponics system.

1. Adhere to ADA Compliance in the design: The most important part of the design was to make sure the handicapped and elderly could use the system. Following ADA compliance would help make sure the system was useable by those who used a wheelchair.
2. Allow for the hydroponics systems to be movable: When designing the system ways to make the system movable was at front thought. Since the system will be in a growing environment such as a green house, floor space is always changing. If a part of the system broke or more systems were brought in, the team wanted to ensure the system could be moved.
3. Allow for ease of use with the hydroponics system: The team wanted the growing aspect of the system to be as easy to use as possible. Allowing ease of access to the crop raceways and water reservoir was sought after in the design.

4. Implement a system using blue tooth: With the system being integrated with the Minka House which already uses Bluetooth for its technology, The team wanted the data that was being taken from the system to be transmitted via blue tooth.

1.4 Issues/Constraints

1. Accessibility and ADA Compliance: The team wants to ensure that the hydroponic system is accessible to all users, addressing potential mobility challenges. We plan to attack this with the design of the system to comply with ADA standards, allowing users with physical abilities to easily use the product.
2. Partial Automation: Full automation of certain processes, such as harvesting and filling the water reservoir, is not feasible with our limit of knowledge, time, and money. We plan to make up for this by focusing on designing these manual processes to be easily accessible and manageable.
3. Accessible water reservoir: Complete automation of the water reservoir filling process is not achievable. We plan to help with this by ensuring the water reservoir is strategically positioned and designed for easy access.
4. User friendly crop raceways: Implement a way to provide easy interaction with the crops. The team plans on achieving this by designing crop raceways at a comfortable height and making the raceways movable.

2.) SMART HYDROPONIC SYSTEM DESIGN & CONSTRUCTION

The physcial design of the project came down to creating one growing bed that can be used in a module fashion within a green house. Depending on the greenhouse's scale, many beds could be placed together, creating an ample growing space. To have the growing bed be able to be used in this fashion it would need to be moved around so creating a lightweight growing bed was also a very important part of the design.

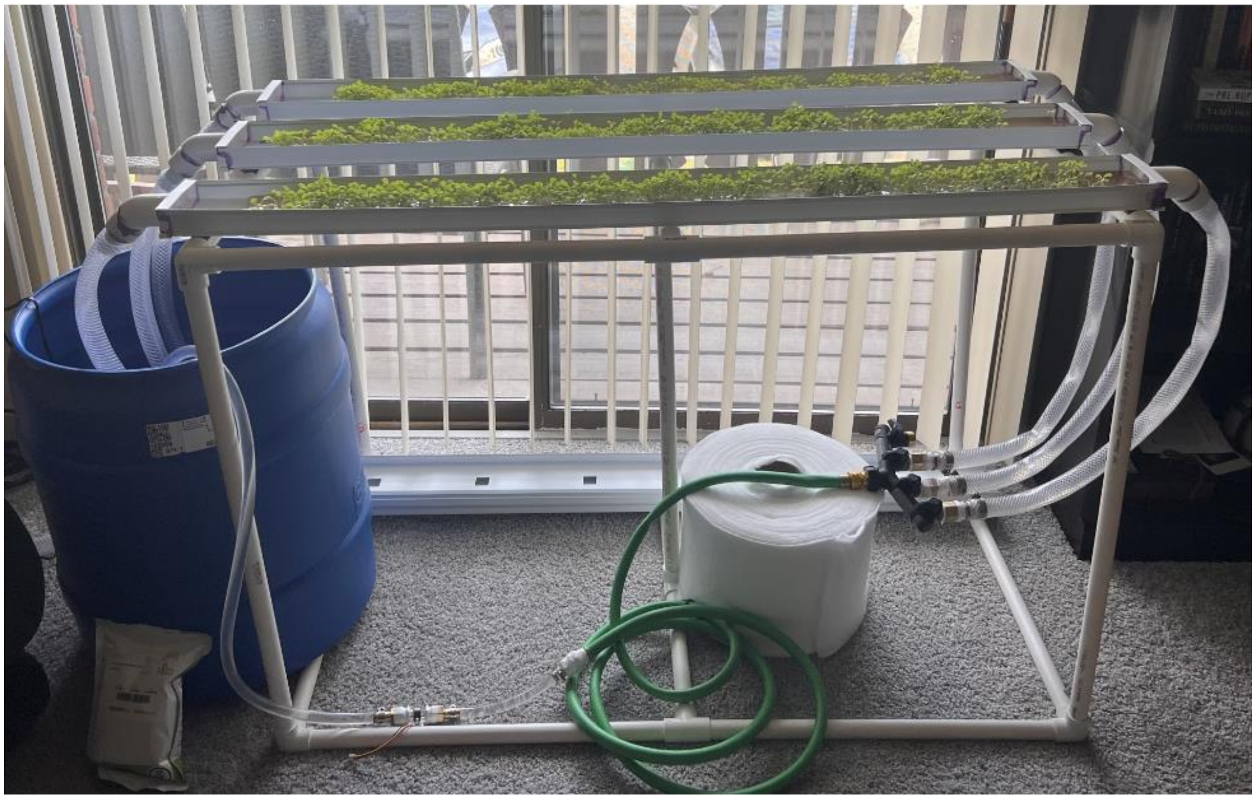


Figure 2: Physical System

2.1 Bed Design

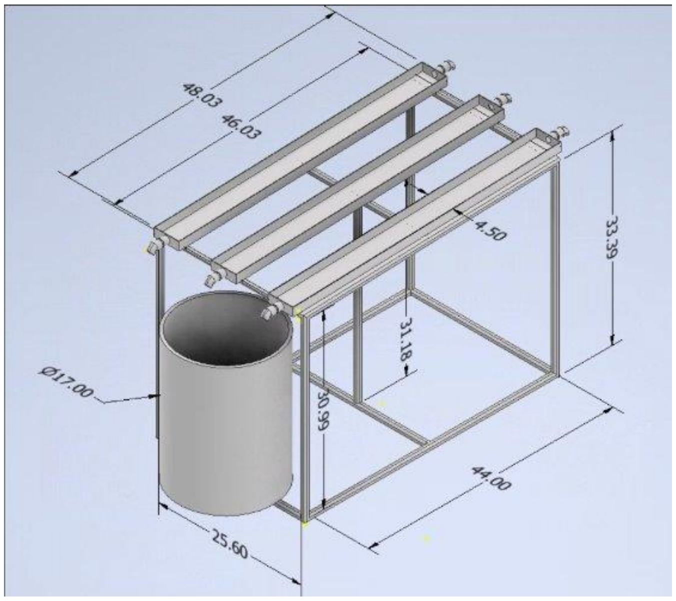


Figure 3: CAD Angle View

The growing bed, seen in Figures 2, 3, & 4 was designed to follow ADA codes seen in the appendix regarding height. This code states that the system could not be over a height of 36”. The height was done this was so users of different physical capabilities may access the system easily. The structure was made of ¾” inch PVC pipe. PVC pipe was chosen so the structure could be changed to fit different situations and expanded on if desired. When dealing with PVC pipe a common handsaw can cut it. So, anyone using the system could take a piece out of the system with ease. PVC pipe would also allow for a user to reconfigure the dimensions of the growing bed for a given space. The growing channels were bought from an outside source measuring 4’ long & 4.5” wide, that is why the length of the system is 44”, this allowed each growing channel to hang off each side allowing easier connections with the piping system. With this, 3 channels could fit comfortably within the width constraint. Also, with a current width of 25.6” it would allow space for an extra channel to be added to the system. The growing channels were secured to the frame of the system using Velcro strips. Securing the channels this way would allow the user to easily take them off if the channels were to break, or to move them on the system if one were to want to add more channels. To allow water flow, the structure was designed to have a 5% grade so that gravity will help circulate the water. A slope of 5% was included so the system would not have to depend on the pump to push the water back into the reservoir and it would allow the system to not easily get clogged by stalled water. Lastly, the tank was placed with half of the opening protruding out underneath the frame. This was chosen because it increased the accessibility to the tank when needing to refill water and adding plant nutrients. Along with the tank's placement, its shape was chosen as round. A round bottom tank is easier to move than a spare or rectangular bottomed tank. For the growth of the crop, a biostrate growth medium was chosen and is placed along the bottom of the three channels of the grow bed, this is essentially the soil for the crops to grow into.

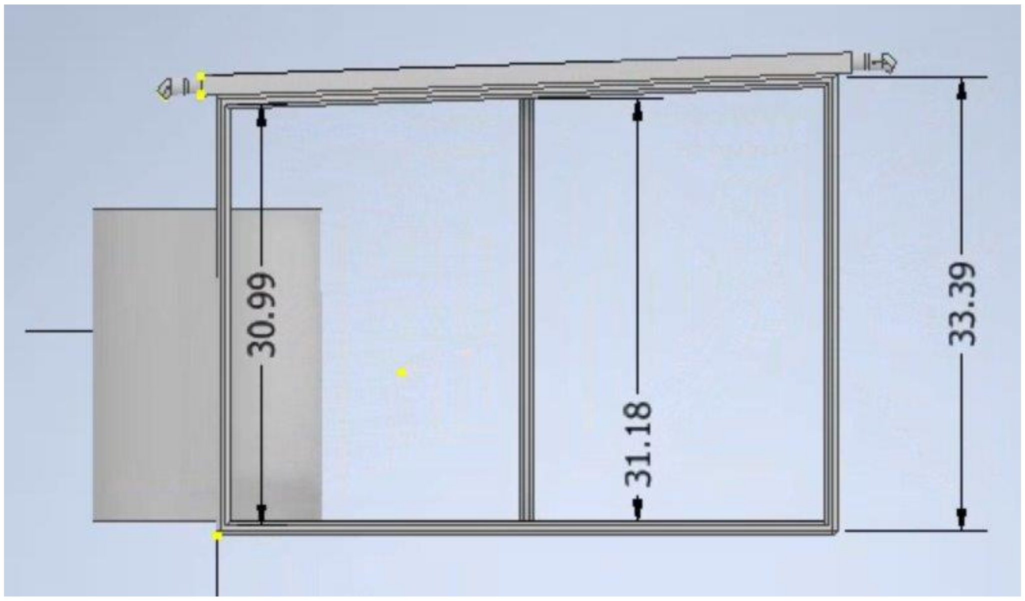


Figure 4: CAD Side View

2.2 Technical Design

As shown in Figure 5, the sensors and pump are placed throughout the system to monitor different aspects of the hydroponics. The pump is placed within the water reservoir and has a variable output of 80-160 GPH. This variable speed was an essential component to the system due to the modularity of the overall project, so it can withstand more channels as well as be dropped down to fewer channels. Since there are three channels in this system requiring around 2 liters/minute, this requires around 95 gallons/hour. So, the variable control on this pump was able to be set to the correct specifications for our system. The water level sensor was placed 3 inches above the pump to give the user plenty of time, around a week with the size of the water tank and depending on the temperature the system is in, to notice the water level readings, so the pump does not get damaged. The Ph sensor simply must be placed within the water reservoir to take consistent readings of the water flowing through the system. And finally, the water flow sensor is placed at a point in the piping so that the sensor can take readings of all the water coming into the system. The water level sensor is a simple float switch, it is coded that when the float mounted on the sensor reaches the bottom of its track, the water level will read low, at all other times the sensor will read high. Next, the flow rate sensor takes constant readings of all water flowing into the system. It is designed to take these readings in liters per minute. This was designed this way due to a hydroponic channel of our size requiring around 2 Liters/min, so this specification can be verified using our water flow sensor. The PH sensor is placed within the water reservoir so that it can take constant measurements of the overall PH levels of the water flowing into the system. This is done because our system uses a liquid nutrient solution, and depending on the plant type being grown, the suggested PH level can vary from 5-8, so this sensor can assist in maintaining an essential nutrient level for the overall health of the plants.

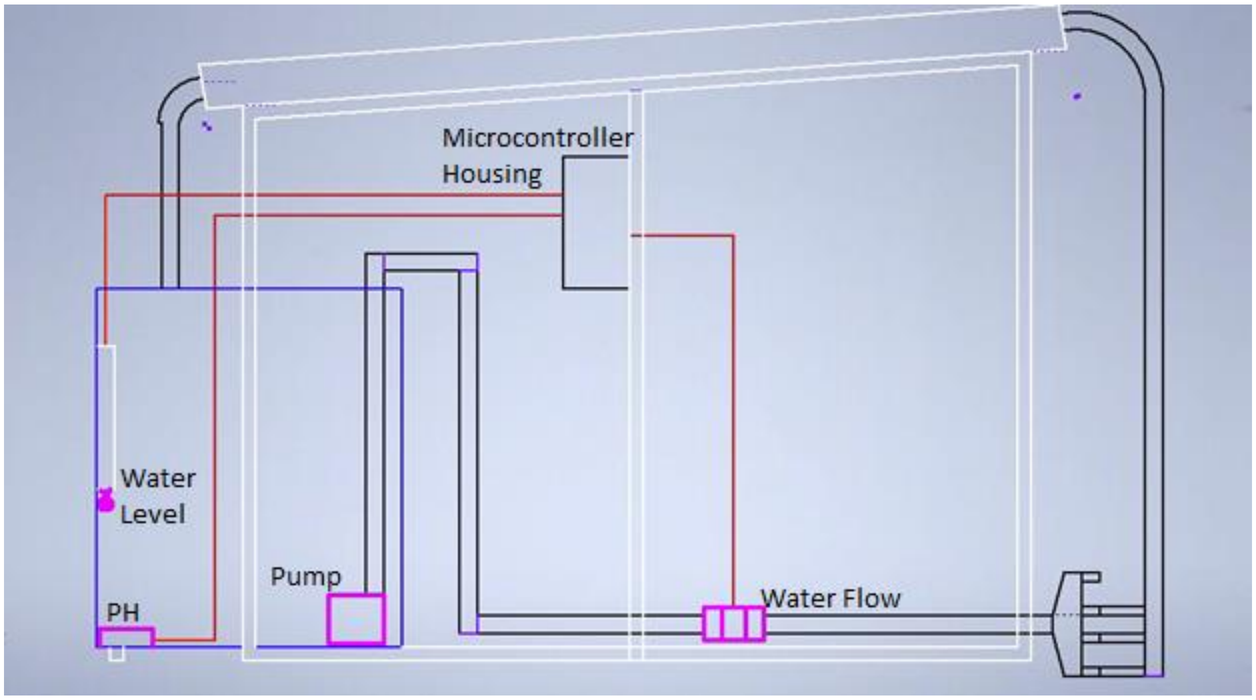


Figure 5: Technical System Layout

2.3 Microcontroller Consideration

Since we want to collect data from our sensors, a microcontroller is needed. It grabs data from sensors, sorts it out, and then sends it over Bluetooth to other devices. Without it, the data wouldn't get where it needs to go smoothly, essentially making it one of the most important parts of our system.

	Name	Cost	Size (in)	Analog Pins	Wi-Fi/Bluetooth
1	**ESP32**	**$9.59**	**1.9x1**	**16**	**Yes**
2	Arduino Micro	$24.50	0.7x1.9	12	No
3	Raspberry Pi 1 Model B+	$25.0	3.35" x 2.2"	0	Yes
4	Arduino UNO	$32.00	2.95x2.1	6	No

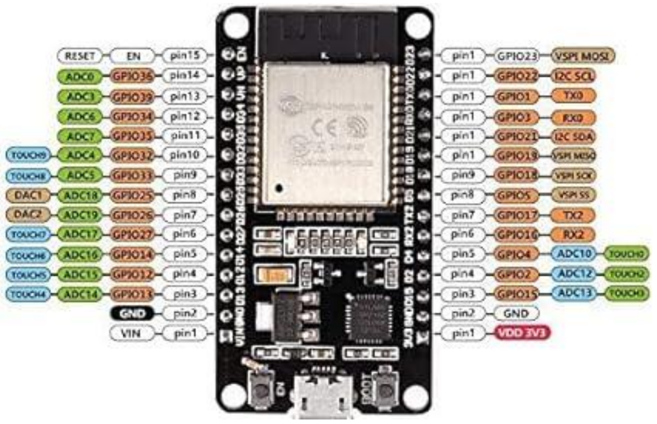


Figure 6: Overall Layout of ESP32 & Design Decision

The ESP32 selected, Figure 6 was the ESP-WROOM-32 ESP32 Development Board. The ESP32 will communicate with an app interface for the user to communicate with the system. This allows for the input of several different sensors that can measure data constantly and apply it to the systems automation and simultaneously send data to the user. One of the main reasons this board was chosen for the design was because of cost and its ability to communicate via blue tooth.

2.4 Flow Rate Sensor

The water flow sensor, Figure 7 chosen was the Clear Turbine Water Flow Sensor with 3-pin JST. The sensor itself is designed to monitor the flow coming from the water reservoir into the channels. A flow rate sensor was necessary because of the risk of blockages, or a pump failure, and the sensor can tell us this if it ever reads 0. It is also a necessary tool because hydroponic systems need a precise amount of water to function at the best level, so the sensor can monitor that for the user and allow them to create a good environment for their system. The sensor design works well for this situation because it can be installed between the reservoir and the channel splits as seen in Figure 5. The installation of the sensor into our system was very straight forward because of its two-sided male threads. Once installed we tested this sensor by maxing out our pump and taking reading from the sensor. The max pump value and sensor reading value were compared, and we confirmed that the sensor was working correctly.

	Name	Cost	ESP32 Compatible	Integration
1	**Adafruit Flow Sensor**	**$6.50**	**Yes**	**Easy**
2	Digiten G3 Sensor	$17.99	Yes	Easy
3	SeedStudio Flow Sensor	$9.50	Yes	Medium
4	MiSUMi	$13.06	No	Easy

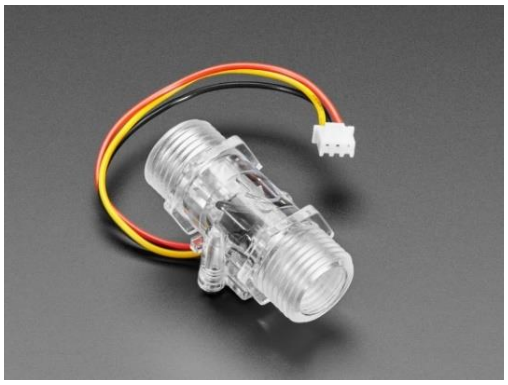
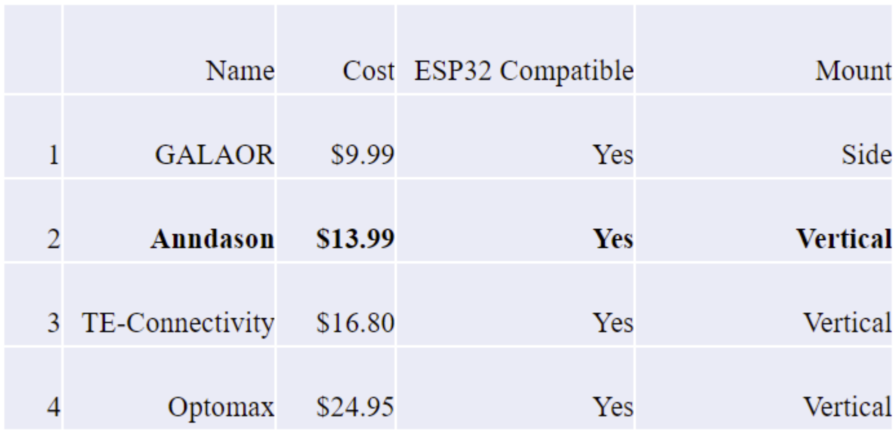


Figure 7: Water Flow Sensor & Design Decision

2.5 Water Level Sensor

The water level sensor chosen was the Anndason 6 Pieces Plastic PP Float Switch. For installation, the sensor was mounted on the end of a PVC pipe and was then mounted within the reservoir, seen in Figure 5, at a height of 3” above the pump so that when the water eventually reaches the level where the float switch activates, an alert will be sent to the user notifying them that the water reservoir needs to be filled. The sensor itself was chosen due to its style of reading, only requiring a single pin and sending in a base HIGH or LOW to the microcontroller. Testing of the sensor was done by simply lowering the float switch into a bowl of water and taking the reading. When it was submerged it read water high. It was then taken out of the water, the switch dropped, and it started reading water low.



	Name	Cost	ESP32 Compatible	Mount
1	GALAOR	$9.99	Yes	Side
2	**Anndason**	**$13.99**	**Yes**	**Vertical**
3	TE-Connectivity	$16.80	Yes	Vertical
4	Optomax	$24.95	Yes	Vertical

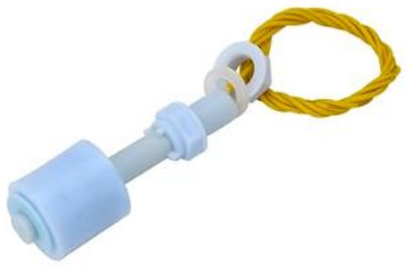
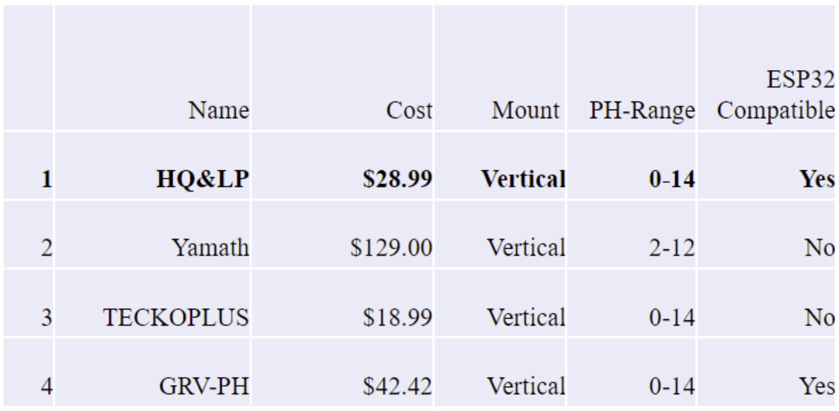


Figure 8: Water Level Sensor & Design Decision

2.6 PH Level Sensor

The final sensor is the PH sensor, the product chosen for this was the HQ&LP PH Sensor, the sensor takes readings of the PH level in the water essentially measuring the nutrient levels form the nutrient packets that will be put into the reservoir. This was a best fit for our system because of its compatibility with arduino IDE as well as the ESP32. Its form factor fit the design needs for this system as well as its ability to read a PH range of 0-14. This was perfect for the design because it would allow for a varaity of plant PH ranges to be grown in the hydroponic system while still being able to monitor PH levels. To test this sensor, a PH test strip was used and placed within our tank as well as placing the PH sensor in the tank. Both values were compared and they matched up so we knew the sensor was working.

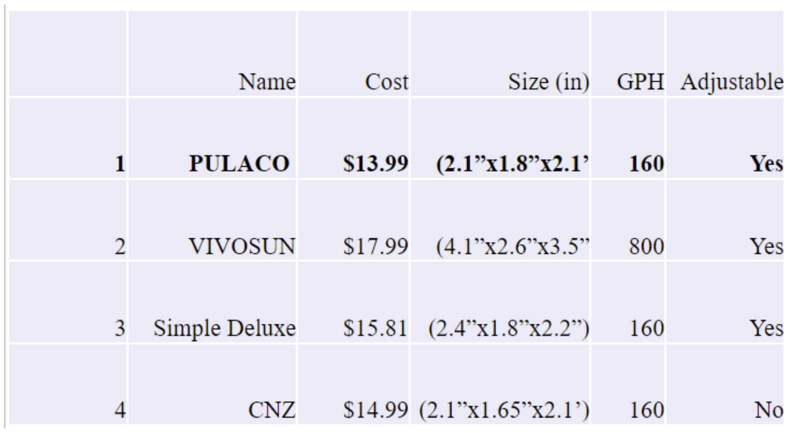


	Name	Cost	Mount	PH-Range	ESP32 Compatible
1	**HQ&LP**	**$28.99**	**Vertical**	**0-14**	**Yes**
2	Yamath	$129.00	Vertical	2-12	No
3	TECKOPLUS	$18.99	Vertical	0-14	No
4	GRV-PH	$42.42	Vertical	0-14	Yes

Figure 9: PH Level Sensor & Design Decision

2.7 Irrigation System

The pump chosen to drive our water system was the PULACO Submersible Pump. The main reason this pump was chosen was for its ability to pump 160GPH and that number can be adjusted down. Another reason this pump was chosen was the fact that it was submersible, so that it can be placed within the water reservoir. In the design, the pump is to be placed within the reservoir with a float sensor to notify the user of water level to ensure the pump does not get exposed. As well as the flow sensor observing water flow coming from the pump to have a constant check for pump failure



	Name	Cost	Size (in)	GPH	Adjustable
1	**PULACO**	**$13.99**	**(2.1”x1.8”x2.1’**	**160**	**Yes**
2	VIVOSUN	$17.99	(4.1”x2.6”x3.5”	800	Yes
3	Simple Deluxe	$15.81	(2.4”x1.8”x2.2”)	160	Yes
4	CNZ	$14.99	(2.1”x1.65”x2.1’)	160	No

Figure 10: Submersible Water Pump

2.8 Wiring Diagram

Seen below Figure 11 is the wiring diagram for the sensors and microcontroller, only three GPIO pins were used to establish communication between the sensors and the microcontroller, leaving room for more sensors and actuators as designed. The power for the microcontroller and sensors came from a wall outlet converted to 3.3 volts using a power supply module placed on the breadboard. The Flow rate sensor is a three-wire sensor, one wire for voltage, one for ground and one communication wire. The water level sensor is a two-wire sensor, one being for ground and the other for communication. And finally, the PH sensor had to be run into an analog to six pin analog to digital converter, but for this system, only three pins were used, one for voltage, one for ground and one for communication.

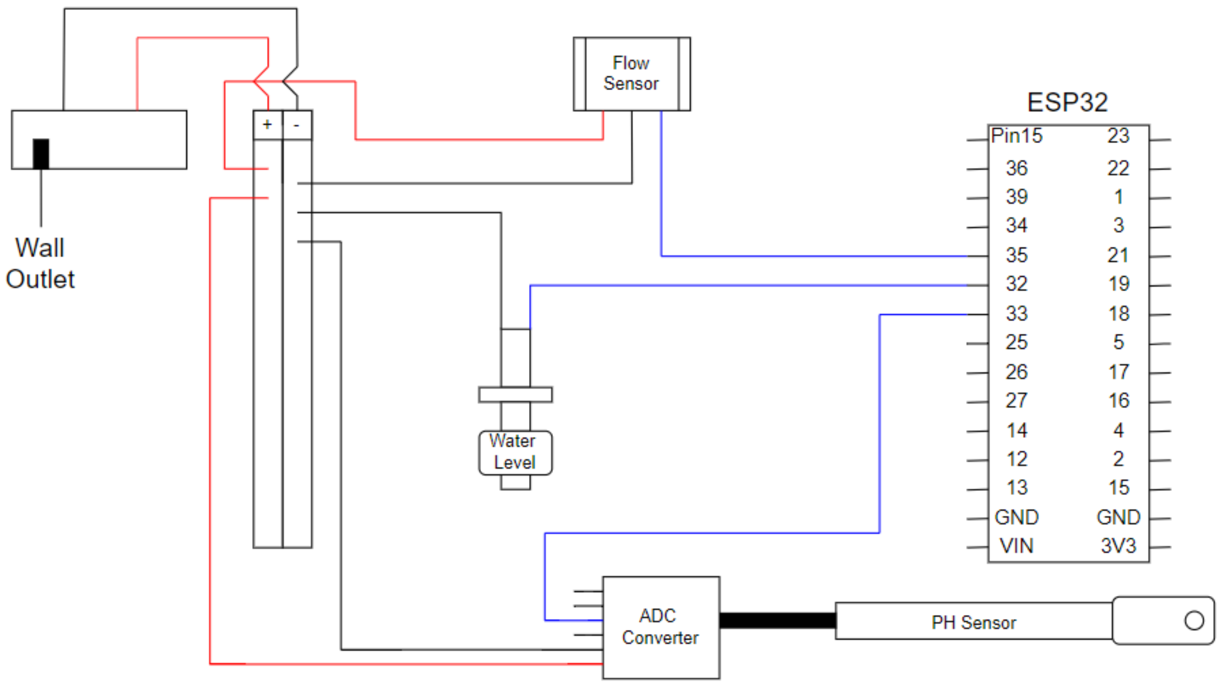


Figure 11: Wiring Diagram

2.7 Microcontroller Programming

To program the microcontroller for the smart hydroponic system, firstly Bluetooth discoverability had to be established. This was done by including the correct libraries for Bluetooth connectivity with for the ESP32 then declaring a name for the ESP32 to be discoverable. Next, because each sensor had a single pin at which they were physically wired, these pins then needed to be declared and set equal to a variable within Arduino IDE accordingly. Beginning with the water level sensor, its readings were taken in by the microcontroller, they were set to a variable called waterLevel which would go through a simple IF ELSE statement due to the sensor sending in a physical HIGH or LOW depending on the float position. Next is the flow rate sensor, when the flow rate sensor readings were taken in by the microcontroller it was set to a variable called flowPulseCount. This variable represented the number of rotations the turbine within the sensor makes per minute. This variable then went through several calculations to convert it to a variable called flowrate representing the speed at which the water is flowing out of the pump in liters/hour. This was put into liters per hour because a single channel for this system needed around 0.5 liters per min, then making the requirement 33 liters per hour. So, for a three-channel system the flow rate needs to be reading around 100 liters per hour consistently. Next is the PH sensor, since the PH sensor must go through an analog to digital converter to get readings the microcontroller is capable of reading, the microcontroller is programmed to take in a value called digitalReading. This variable is the result of tuning done physically on the analog to digital converter. But more calculations and conversions had to be done within Arduino IDE to get a value representing an accurate PH reading of the water in the system. After the three sensors are communicating and outputting values that are true to the system, they are then transmitted as a single string of readings over Bluetooth. This allows for when the app connects to the ESP32 via Bluetooth, it can take in the sensor readings, break the string of values up, and assign them to locally declared variables accordingly.

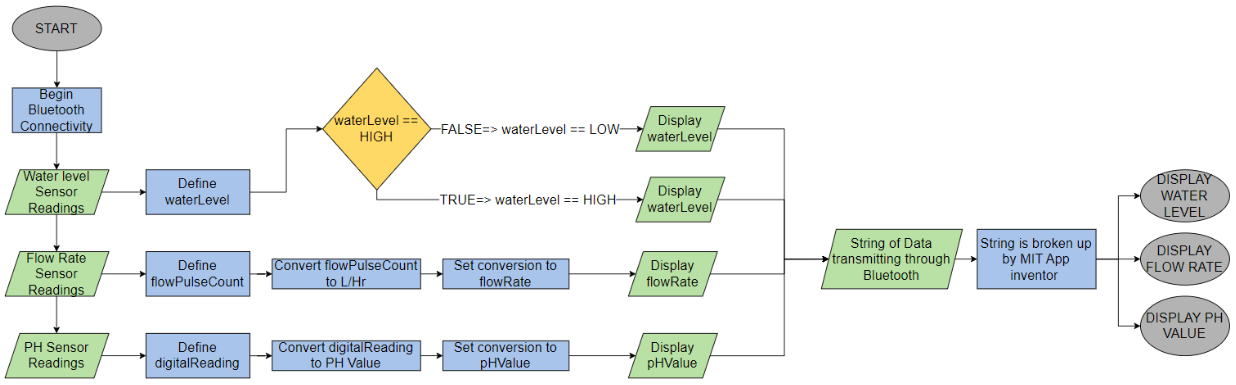
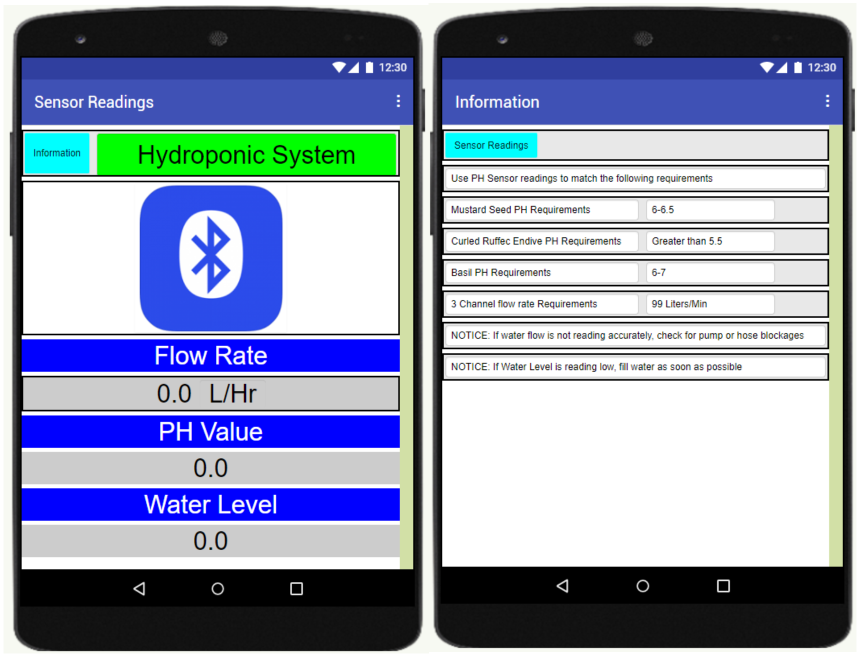

START
Begin Bluetooth Connectivity
Water level Sensor Readings
Define waterLevel
waterLevel == HIGH
FALSE=> waterLevel == LOW
Display waterLevel
TRUE=> waterLevel == HIGH
Display waterLevel
Flow Rate Sensor Readings
Define flowPulseCount
Convert flowPulseCount to L/Hr
Set conversion to flowRate
Display flowRate
PH Sensor Readings
Define digitalReading
Convert digitalReading to PH Value
Set conversion to pHValue
Display pHValue
String of Data transmitting through Bluetooth
String is broken up by MIT App inventor
DISPLAY WATER LEVEL
DISPLAY FLOW RATE
DISPLAY PH VALUE

Figure 12: Microcontroller Programming

2.8 User Interface - App



Use PH Sensor readings to match the following requirements

Mustard Seed PH Requirements	6-6.5
Curled Ruffec Endive PH Requirements	Greater than 5.5
Basil PH Requirements	6-7
3 Channel flow rate Requirements	99 Liters/Min

NOTICE: If water flow is not reading accurately, check for pump or hose blockages

NOTICE: If Water Level is reading low, fill water as soon as possible

Figure 12: User Interface – App & Example of Programming Style

The chosen user interface was an app communicating with the ESP32 through Bluetooth. This was achieved by using a program called MIT app inventor, which allows for simple communication with an ESP32. This app displays the three sensor readings for the user and can be adapted internally to set its flags for insufficient water flow and PH levels depending on the choice of crop. This app displays whether the water level sensor is reading high or low, constant readings of the water flow coming through the water flow sensor in liters per minute, and a constant reading of the PH level within the water reservoir. When monitoring the app, if a sensor is reading a value that is not sufficient for the system, it will flag this reading prompting the user to perform the specified action on the system. A second screen was designed to display different information about target numbers for the system, including recommended PH levels for different microgreens and the target readings for the flow rate and water level sensor. To program the app, MIT app inventor uses a coding style similar to Scratch, an introductory coding style used to introduce users to the thought process behind coding logic. To use this logic-based code to display the three sensor readings, several steps must be taken. The app itself must first be capable of discovering Bluetooth devices. To do this, a Bluetooth client had to be installed onto the software from an external source. Utilizing this Bluetooth client, the app was designed for when the user clicks on the Bluetooth icon, it then brings up the ESP32 searching for a connection. After this was completed, the next step was to declare variables within the apps software for each of the three sensors. The code programmed onto the microcontroller is designed so that it outputs the sensor readings as a single string representing all three sensor readings. The app then takes in this string and assigns it to what is called a list in MIT App Inventor. This list is then broken up into three separate values by searching through the list for a single comma, this is how the program knows when one reading ends, and the next begins. Once the string is properly broken up, each value is then assigned to its respective local variable that was previously declared in the apps program. Now that these readings are properly dispersed in the program, the screens can now be designed to display these readings accordingly.

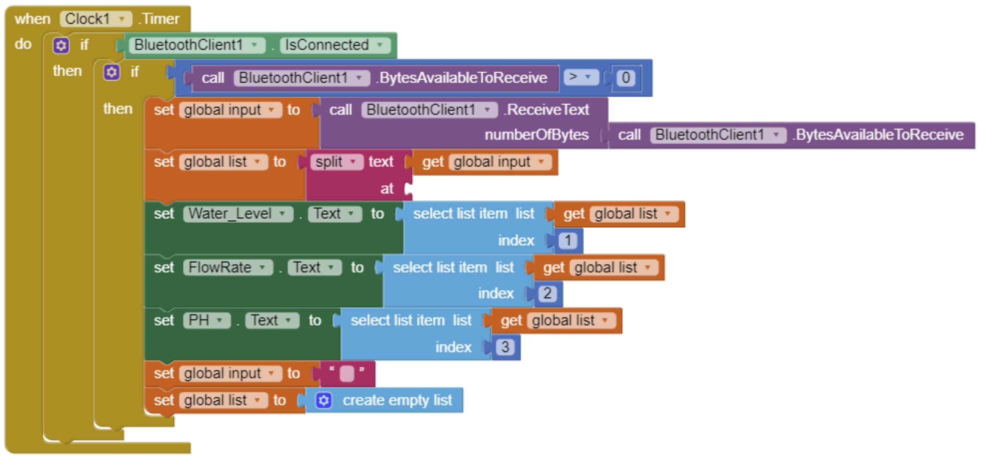

```
when Clock1 .Timer
do  if  BluetoothClient1 . IsConnected
    then  if  call BluetoothClient1 .BytesAvailableToReceive  >  0
          then  set global input to  call BluetoothClient1 .ReceiveText
                                          numberOfBytes  call BluetoothClient1 .BytesAvailableToReceive
                set global list to  split  text  get global input
                                           at
                set Water_Level . Text to  select list item  list  get global list
                                                             index  1
                set FlowRate . Text to  select list item  list  get global list
                                                          index  2
                set PH . Text to  select list item  list  get global list
                                                    index  3
                set global input to  " "
                set global list to  create empty list
```

Figure 13: Example of MIT App Inventor Code

2.9 App Integration

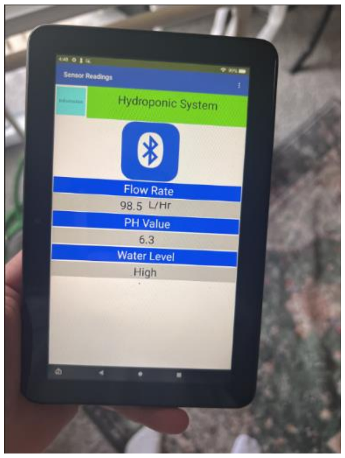


Figure 14: App Integration

In order to connect to the app on a handheld device, the user must first scan a QR code to get the programmed app onto their device. Once this is done, the user only needs to establish the Bluetooth connection by clicking on the Bluetooth icon displayed on their mobile device. Once this connection is established, the user will immediately begin to receive constant readings about the hydroponic system. The app will appear as seen below Figure 14. The app is currently being used and tested using an amazon fire 7, however this can be easily adjusted to function accordingly on devices utilized by the Minka House residents.

2.10 Building Process

The building process is based around the implementation of one building block at a time. The hydroponic system itself was first designed. Then a physical structure was built as well as the water system. Then when the whole system is functioning properly without any technology the sensors can then be implemented so proper readings can be taken. Once these sensors are communicating properly with the microcontroller the app will be developed so that the ESP32 can be disconnected from the computer and communicate the readings to the app alone. This thought process is displayed below (Figure 10)

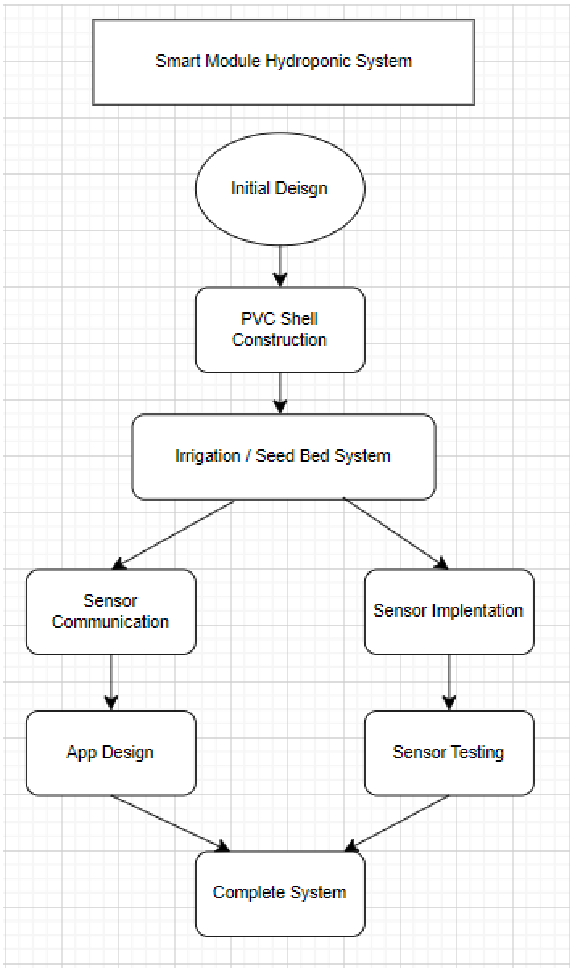


Figure 15: Building Process Flow Chart

2.11 Teamwork Breakdown

The design is broken up into 5 subsystems, and the teamwork breakdown is planned as displayed below. With Gavin working on design the frame of the system using PVC pipe and determining the length & width of the channels, the sensor implantation of the water flow & water level sensors, the tank & piping system, and lastly the App. Also, Tyler working on the sensor communication for all three sensors, the implantation of the PH sensor, serial communication and APP, lastly the pump.

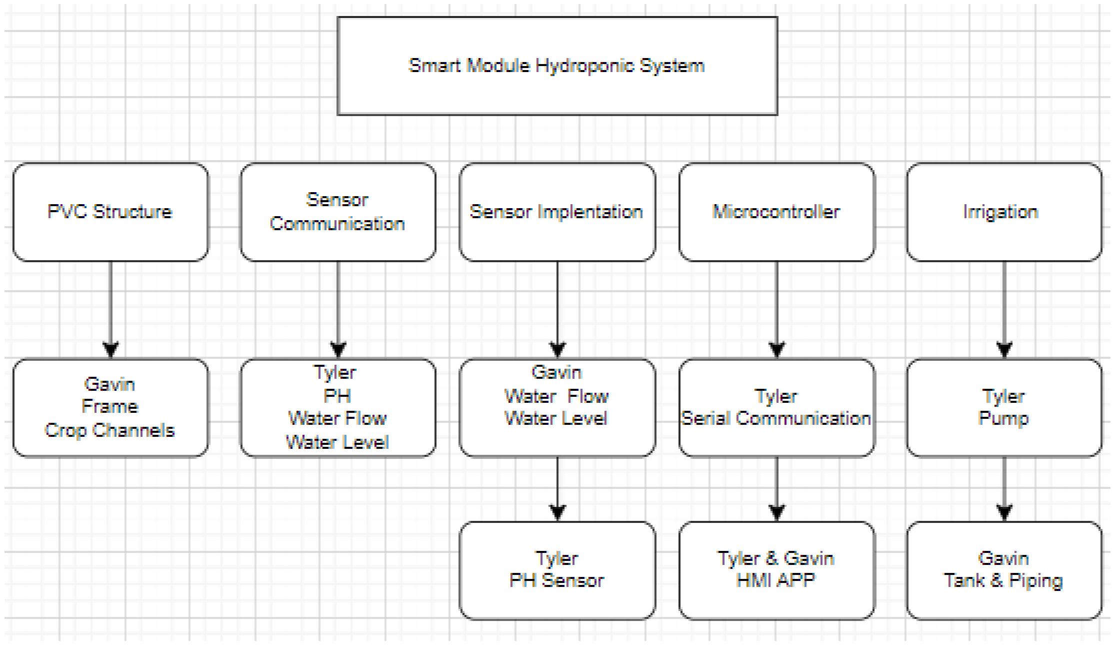


Figure 16: Teamwork Breakdown

2.12: Project Breakdown

Seen below is the project block diagram, describing all parts and systems that contribute to a finished hydroponic system. Beginning with the hardware, the sensors being a PH, water flow, and water level. Beginning with PH, this sensor will measure the PH level in the water going through the plants. The user can use the app to ensure that that the PH level does not go under their specified threshold. Next, the water flow sensor will be placed on the main pipe running from the water reservoir taking readings of the water flow. This sensor's purpose is purely to ensure water flow stays above a certain level, and if it goes below the set number, it will prompt the user to check for blockages or pump failure. Lastly, is the water level sensor, this float sensor will be mounted within the water reservoir to ensure that water level never reaches a level where it exposes the water pump, potentially damaging it. If the water level goes below a certain level, the float sensor will prompt the user to refill the water reservoir. Another portion of the hardware of the system is the ESP32. It is the blue tooth microcontroller communicating the readings from the sensors to the user interface. The final piece of hardware is the pump, the pump just creates a constant circulation of water throughout the system. Next is the software, the program used to code the microcontroller was Arduino 2.0 IDE, chosen due to its extensive library support for the esp32, as well prior experience using this software. The user interface chosen is an app connected by Bluetooth to the ESP32. To achieve this, MIT app inventor was chosen so the user can easily implement on different devices to monitor their system.

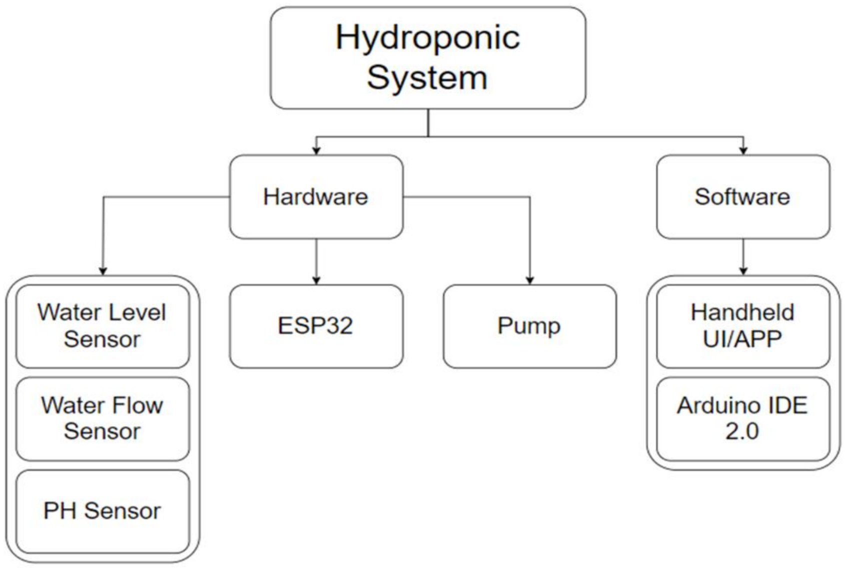
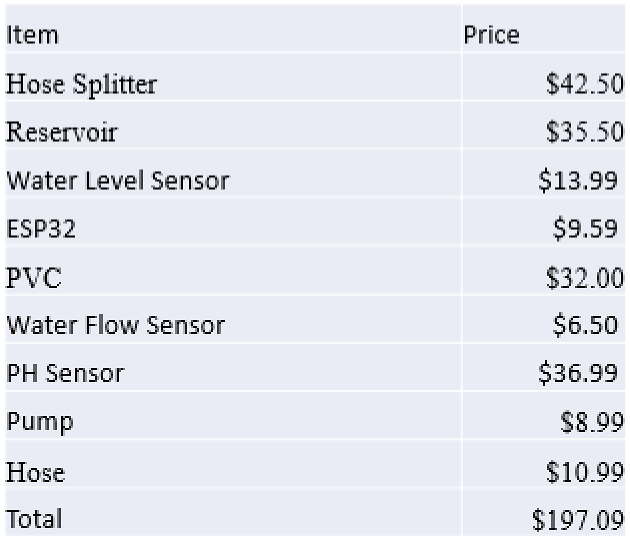


Figure 17: Project Breakdown

3.) ECONOMIC CONSIDERATIONS

3.1 Bill of Materials for Construction

This shows the cost of physical building the system.

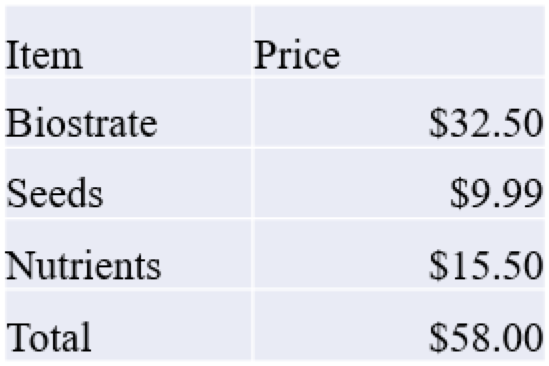


Item	Price
Hose Splitter	$42.50
Reservoir	$35.50
Water Level Sensor	$13.99
ESP32	$9.59
PVC	$32.00
Water Flow Sensor	$6.50
PH Sensor	$36.99
Pump	$8.99
Hose	$10.99
Total	$197.09

Figure 17 – System Build Cost

3.2 Recuring Cost

Figure 18 – System Recurring Cost



Item	Price
Biostrate	$32.50
Seeds	$9.99
Nutrients	$15.50
Total	$58.00

This shows the cost of running the team's system. This price could be subject to change depending on the product choice for the user's choices for their system.

4.) CONCLUSIONS AND RECOMMENDATIONS

4.1 Current Progress

With the current standing of the project, our team has designed a single fully functional hydroponic grow bed with a fully monitored watering system using multiple sensors. With this design, users will be able to expand upon it with little additional design and will still be able to monitor the different aspects of their entire system due to design choices.

4.2 Future Recommendations

The following issues are left open for future engineering students who want to improve upon the completion of this project:

- Expansion – With this designed system, it can be expanded upon easily and still have the same interaction with the app. So, a user can integrate multiple systems with little additional design needed.
- More automation – Trying to find a way that cuts out almost all user interaction such as designing automatic seeding / harvesting actuator.

- Controlled climate – If a whole green house is implemented in the future, finding a way to include heaters, fans, lights, and ventilation would go a long way into making this hydroponic system into a truly smart system.

4.3 Reflection

Creating a smart hydroponic system was a challenging yet rewarding project. Designing the system and creating the code for the App took a lot of time and creativity but seeing it in action and witnessing the first microgreens we planted, sprout, brought a sense of accomplishment to the team. It's exciting to think about the potential impact this hydroponic system could have on the Minka house and personal use.

References

K. Nemali, “History of Controlled Environment Horticulture: Greenhouses,” HortScience, vol. 57, no. 2, pp. 239–246, Feb. 2022, doi: https://doi.org/10.21273/HORTSCI16160-21.

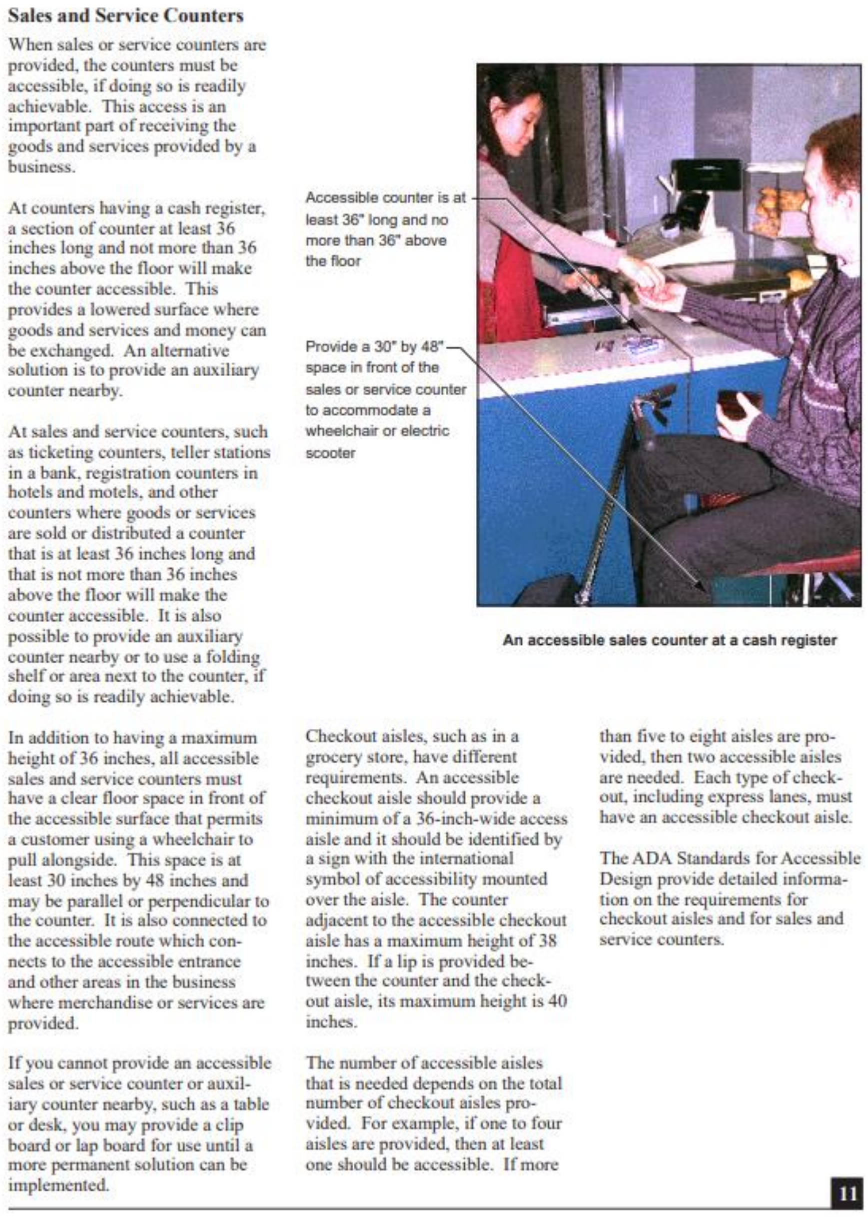
“What is Hydroponic system? Everything you need to know - Mission Sustainability,” Jul. 10, 2022. https://www.missionsustainability.org/blog/what-is-hydroponic-system#:~:text=The%20Aztecs%20and%20the%20Chinese (accessed Oct. 26, 2023).

“The History Of Hydroponics,” Two Wests & Elliott Ltd. https://www.twowests.co.uk/a/blog/the-history-of-hydroponics#:~:text=In%201924%20Dr%20William%20F

R. Holmer, G. Linwattana, P. Nath, and J. D. H. Keatinge, SEAVEG 2012: High Value Vegetables in Southeast Asia: Production, Supply and Demand. AVRDC-WorldVegetableCenter, 2013. Accessed: Oct. 26, 2023. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=rMrUsJIOPj4C&oi=fnd&pg=PA216&dq=microgreen+history&ots=u4nNLNj4ZB&sig=AyNRaW8e-mK_F5bKyAy3r7BG1uE#v=onepage&q=microgreen%20history&f=false

Appendix

Appendix A: ADA Compliance Standards



Sales and Service Counters

When sales or service counters are provided, the counters must be accessible, if doing so is readily achievable. This access is an important part of receiving the goods and services provided by a business.

At counters having a cash register, a section of counter at least 36 inches long and not more than 36 inches above the floor will make the counter accessible. This provides a lowered surface where goods and services and money can be exchanged. An alternative solution is to provide an auxiliary counter nearby.

At sales and service counters, such as ticketing counters, teller stations in a bank, registration counters in hotels and motels, and other counters where goods or services are sold or distributed a counter that is at least 36 inches long and that is not more than 36 inches above the floor will make the counter accessible. It is also possible to provide an auxiliary counter nearby or to use a folding shelf or area next to the counter, if doing so is readily achievable.

Accessible counter is at least 36" long and no more than 36" above the floor
Provide a 30" by 48" space in front of the sales or service counter to accommodate a wheelchair or electric scooter

An accessible sales counter at a cash register

In addition to having a maximum height of 36 inches, all accessible sales and service counters must have a clear floor space in front of the accessible surface that permits a customer using a wheelchair to pull alongside. This space is at least 30 inches by 48 inches and may be parallel or perpendicular to the counter. It is also connected to the accessible route which connects to the accessible entrance and other areas in the business where merchandise or services are provided.

If you cannot provide an accessible sales or service counter or auxiliary counter nearby, such as a table or desk, you may provide a clip board or lap board for use until a more permanent solution can be implemented.

Checkout aisles, such as in a grocery store, have different requirements. An accessible checkout aisle should provide a minimum of a 36-inch-wide access aisle and it should be identified by a sign with the international symbol of accessibility mounted over the aisle. The counter adjacent to the accessible checkout aisle has a maximum height of 38 inches. If a lip is provided between the counter and the checkout aisle, its maximum height is 40 inches.

The number of accessible aisles that is needed depends on the total number of checkout aisles provided. For example, if one to four aisles are provided, then at least one should be accessible. If more than five to eight aisles are provided, then two accessible aisles are needed. Each type of checkout, including express lanes, must have an accessible checkout aisle.

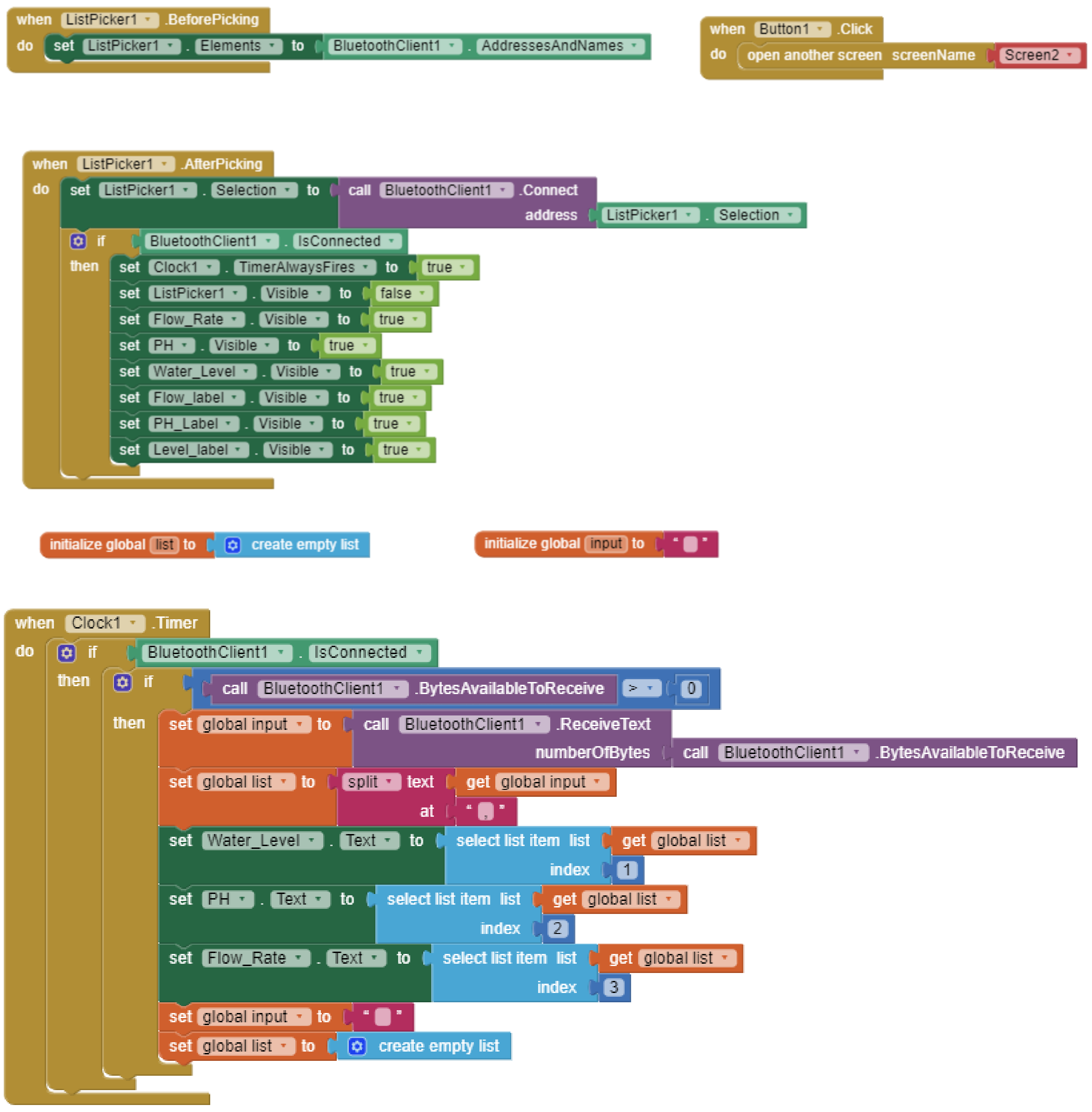
The ADA Standards for Accessible Design provide detailed information on the requirements for checkout aisles and for sales and service counters.

11

Appendix B: Complete Code

```
// init Class:
//BluetoothSerial ESP_BT;

#include <BluetoothSerial.h>
#include <Arduino.h>
#include <Wire.h>

BluetoothSerial SerialBT;

//#define PH_SENSOR_PIN 34 // Analog pin connected to the pH sensor
#define OFFSET_VOLTAGE 3918.0 // Offset voltage of the sensor
#define PH_VOLTAGE_CONSTANT 0.0185 // Voltage to pH conversion constant
#define FLOW_SENSOR_PIN 35  // Define the pin connected to the flow sensor
#define FLOAT_SWITCH_PIN 32   // Pin connected to the float switch
#define SENSOR_PIN 36  // Analog pin connected to the pH sensor


volatile int flowPulseCount = 0;
float flowRate = 0;
unsigned int flowMilliLitres = 0;
unsigned long totalMilliLitres = 0;
unsigned long oldTime = 0;
const int floatSwitchPin = 34; // Define the pin connected to the float switch

void setup() {
  SerialBT.begin("ESP32_Control"); //Name of your Bluetooth interface -> will
show up on your phone
  Serial.begin(9600); // Initialize serial communication for debugging
  pinMode(FLOW_SENSOR_PIN, INPUT_PULLUP); // Set the flow sensor pin as input
with internal pull-up resistor
  attachInterrupt(digitalPinToInterrupt(FLOW_SENSOR_PIN), flowPulseCounter,
FALLING); // Attach interrupt to the flow sensor pin
  pinMode(FLOAT_SWITCH_PIN, INPUT_PULLUP); // Set the float switch pin as input
with internal pull-up resistor
  pinMode(SENSOR_PIN, INPUT);
}

void loop() {
  unsigned long currentTime = millis();

  // Every second, calculate flow rate and total volume
  if ((currentTime - oldTime) > 1000) {
    detachInterrupt(digitalPinToInterrupt(FLOW_SENSOR_PIN)); // Disable the
interrupt
    flowRate = ((1000.0 / (currentTime - oldTime)) * flowPulseCount) / 7.5; //
7.5 is the calibration factor for this sensor
    oldTime = currentTime;
    flowMilliLitres = (flowPulseCount * 60) / 7.5; // Convert pulse count to flow
in millilitres
    totalMilliLitres += flowMilliLitres;
    flowPulseCount = 0;
    attachInterrupt(digitalPinToInterrupt(FLOW_SENSOR_PIN), flowPulseCounter,
FALLING); // Enable the interrupt

    int waterLevel = digitalRead(FLOAT_SWITCH_PIN); // Read the state of the
float switch

  if (waterLevel == HIGH) {
    Serial.println("Water level is high"); // Print a message if water level is
high
    waterLevel = 1;
    SerialBT.println("High"); // Send the sensor value over Bluetooth
    SerialBT.println(",");
  } else {
    Serial.println("Water level is low"); // Print a message if water level is
low
    waterLevel = 0;
    SerialBT.println("Low"); // Send the sensor value over Bluetooth
     SerialBT.println(",");
  }


  float pHValue = readpH(); // Read pH value from sensor
  Serial.print("pH Value: ");
  Serial.println(pHValue); // Print pH value to serial monitor
  SerialBT.println(pHValue); // Send the sensor value over Bluetooth
  SerialBT.println(",");
  }

  // Print flow rate and total volume
  Serial.print("Flow Rate: ");
  Serial.print(flowRate);
  SerialBT.println(flowRate); // Send the sensor value over Bluetooth
 //  SerialBT.println(",");
  Serial.print(" L/min - ");
  //SerialBT.println(flowRate); // Send the sensor value over Bluetooth
  delay(1000); // Add a small delay for stability

}

void flowPulseCounter() {
  flowPulseCount++;
}

float readpH() {
  float voltage = analogRead(SENSOR_PIN) * (5.0 / 1024.0); // Read voltage
  float pHValue = (voltage - OFFSET_VOLTAGE) / PH_VOLTAGE_CONSTANT; // Convert
voltage to pH

  // Convert voltage to pH value using calibration parameters
  // You need to calibrate the sensor using known pH solutions to get accurate
readings
  // This is just a placeholder conversion formula
  pHValue = (voltage - OFFSET_VOLTAGE) / PH_VOLTAGE_CONSTANT; // Convert voltage
to pH/* Your conversion formula using sensor readings */;

  return pHValue;
```

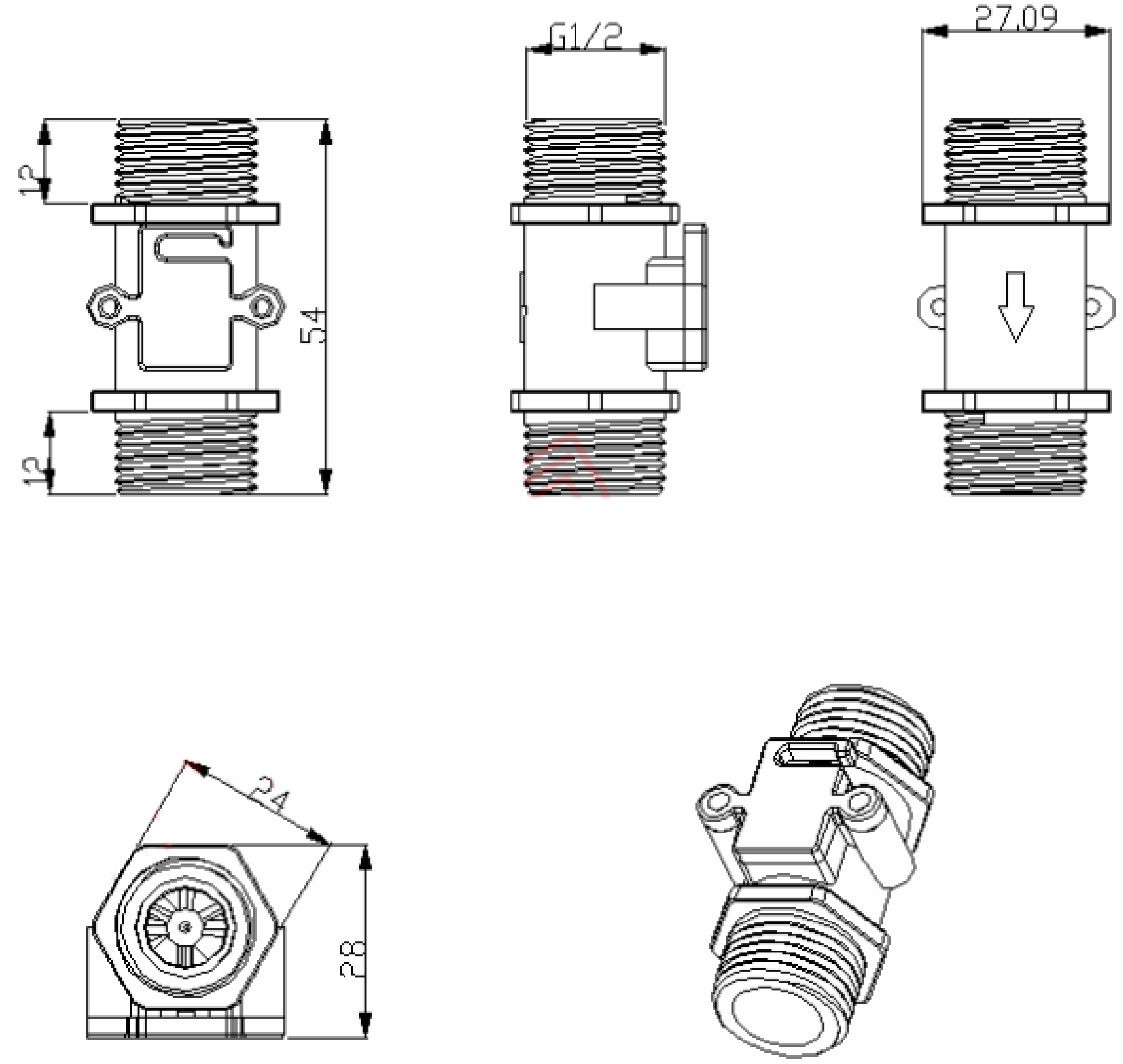
Appendix C: MIT app inventor scratch code



```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to  BluetoothClient1 . AddressesAndNames

when Button1 .Click
do  open another screen  screenName  "Screen2"

when ListPicker1 .AfterPicking
do  set ListPicker1 . Selection to  call BluetoothClient1 .Connect
                                                    address  ListPicker1 . Selection
    if  BluetoothClient1 . IsConnected
    then  set Clock1 . TimerAlwaysFires to  true
          set ListPicker1 . Visible to  false
          set Flow_Rate . Visible to  true
          set PH . Visible to  true
          set Water_Level . Visible to  true
          set Flow_label . Visible to  true
          set PH_Label . Visible to  true
          set Level_label . Visible to  true

initialize global list to  create empty list

initialize global input to  " "

when Clock1 .Timer
do  if  BluetoothClient1 . IsConnected
    then  if  call BluetoothClient1 .BytesAvailableToReceive  >  0
          then  set global input to  call BluetoothClient1 .ReceiveText
                                                    numberOfBytes  call BluetoothClient1 .BytesAvailableToReceive
                set global list to  split text  get global input
                                          at  ","
                set Water_Level . Text to  select list item  list  get global list
                                                            index  1
                set PH . Text to  select list item  list  get global list
                                                  index  2
                set Flow_Rate . Text to  select list item  list  get global list
                                                         index  3
                set global input to  " "
                set global list to  create empty list
```

Appendix D: Cut Sheets

Water Flow Sensor Dimensions



Water Level Sensor

Float level switch is a simple structure, easy to use level control device, it does not provide power, no complicated circuit, it has a mechanical switch than the average small size, long life and other advantages. As long as the material used in the correct shape and nature of any liquid or pressure, temperature can be used, which in the shipbuilding industry, generating equipment, petrochemical industry, food industry, water treatment equipment, dyeing and finishing industry, hydraulic machinery, etc. have been widely application.

How it works: liquid level sensor magnetic reed switch by having an important and integral float, float within the magnetic material in a sealed non-magnetic metal pipe or plastic pipe set one or more magnetic reed switch, and then the catheter through a or more with magnetic float, and double-loop control using a fixed float and reed switch in the relevant position, the float rises or falls with the liquid using the ball close to the beginning of the reed contacts, resulting in open and shut action for level control or direction (when the float near the reed switch is turned on; leave the switch off)

Max Contact Rating :10W

Max Switching Voltage:220V

Max Switching Current:0.5A

Max Breakdown Voltage:220VDC

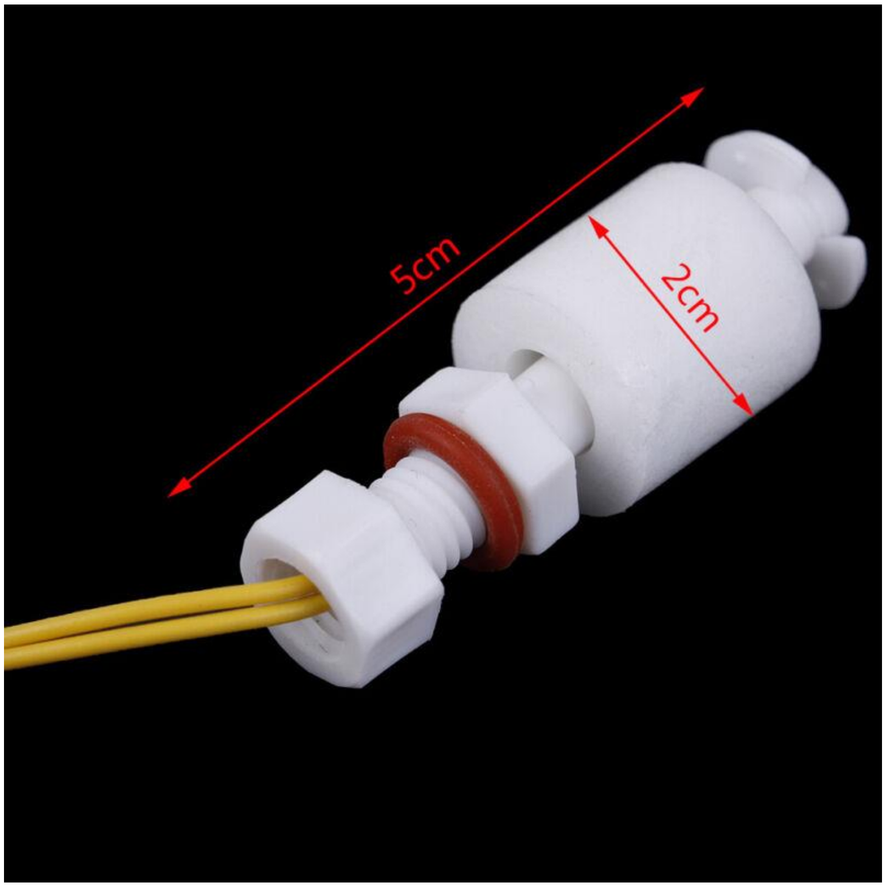
Max Carry Current:1.0A

Max Contact Resistance:100mohm

Temperature Rating:-10~+60°

Float Ball Material:P.P

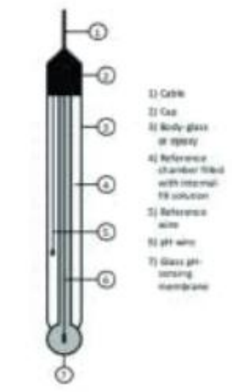
Float Body Matertal:P.P


5cm
2cm

Ph Level Sensor

How pH probe work electronically?

A pH probe consists of two main parts: a glass electrode and a reference electrode as shown in the picture below. In the nutshell, pH is determined essentially by measuring the voltage difference between these two electrodes. The pH probe is a passive sensor, which means no excitation voltage or current is required. It produces a voltage output that is linearly dependent upon the pH of the solution being measured. An ideal pH probe produces 0v output when pH value is at 7, and it produces a positive voltage (a few hundred mili-volts) when pH value go down, and a negative voltage level when pH value go up, causing by the hydrogen irons forming at the outside (and inside) of the membrane glass tip of the pH probe when the membrane comes into contact with solution. The source impedance of a pH probe is very high because the thin glass bulb has a large resistance that is typically in the range of 10 MΩ to 1000 MΩ. Whatever measurement circuit connect to the probe requires to be high-impedance in order to minimise the loading effect of the circuit.



1) Cable
2) Cap
3) Body-glass or epoxy
4) Reference chamber filled with internal fill solution
5) Reference wire
6) pH wire
7) Glass pH-sensing membrane

Figure 1. Typical pH Glass Electrode

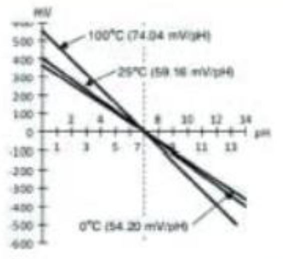


Figure 2. pH-Electrode Transfer Function

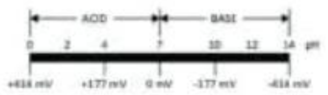
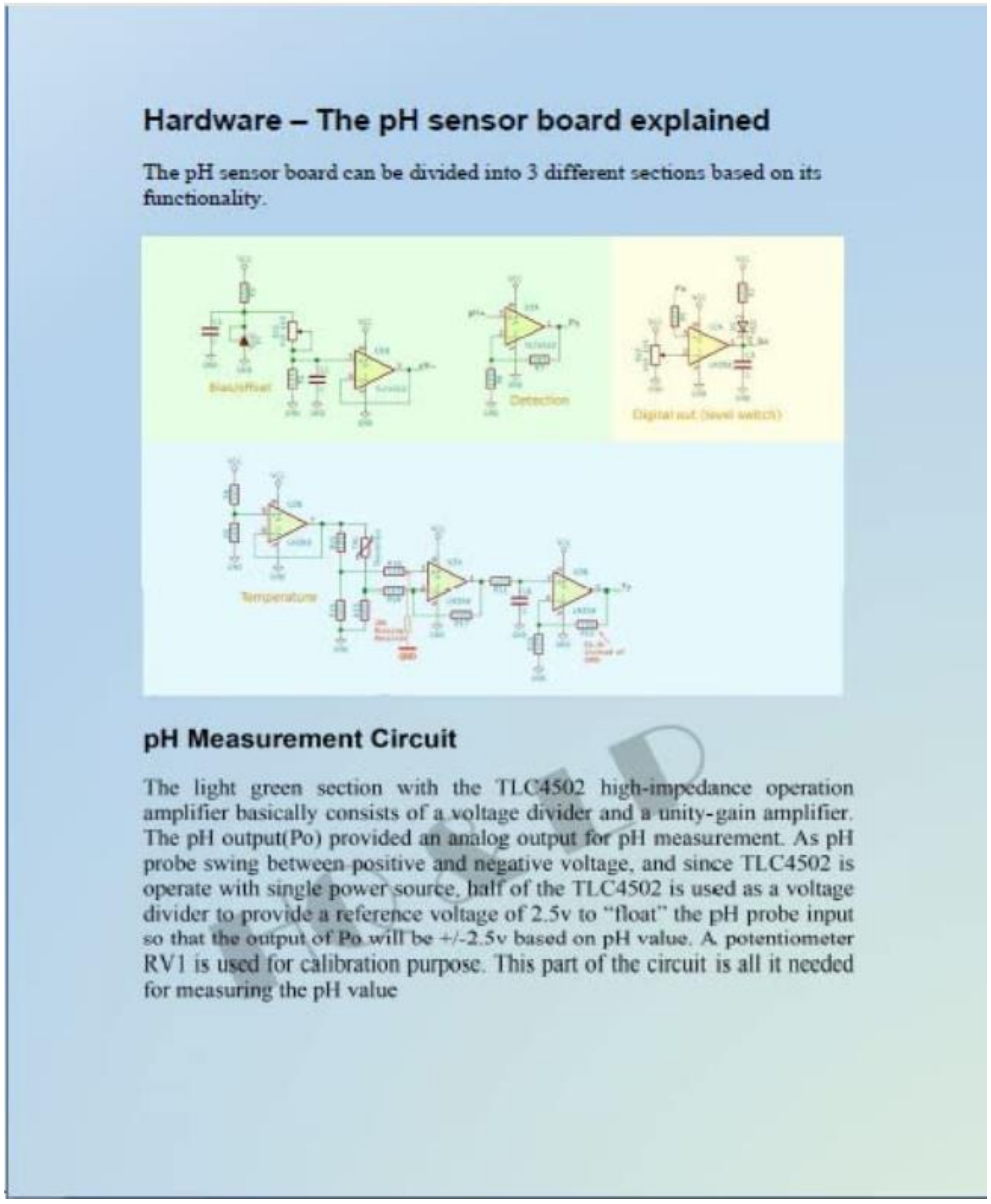


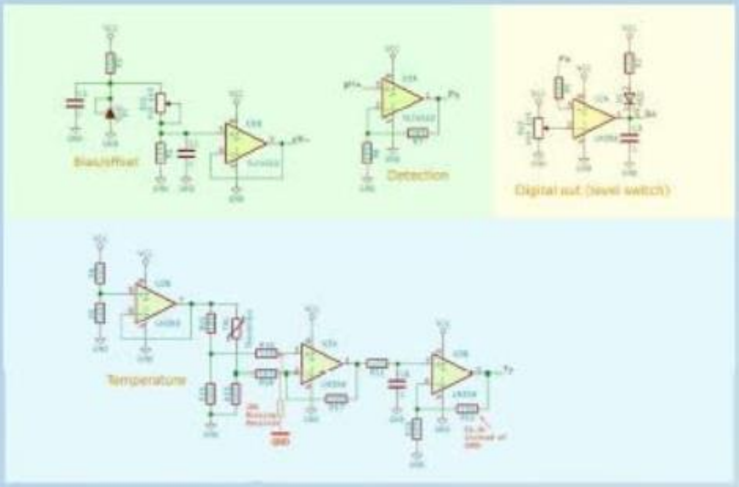
Figure 3. pH Scale

Source: AN-1852 Designing With pH Electrodes, Texas Instruments



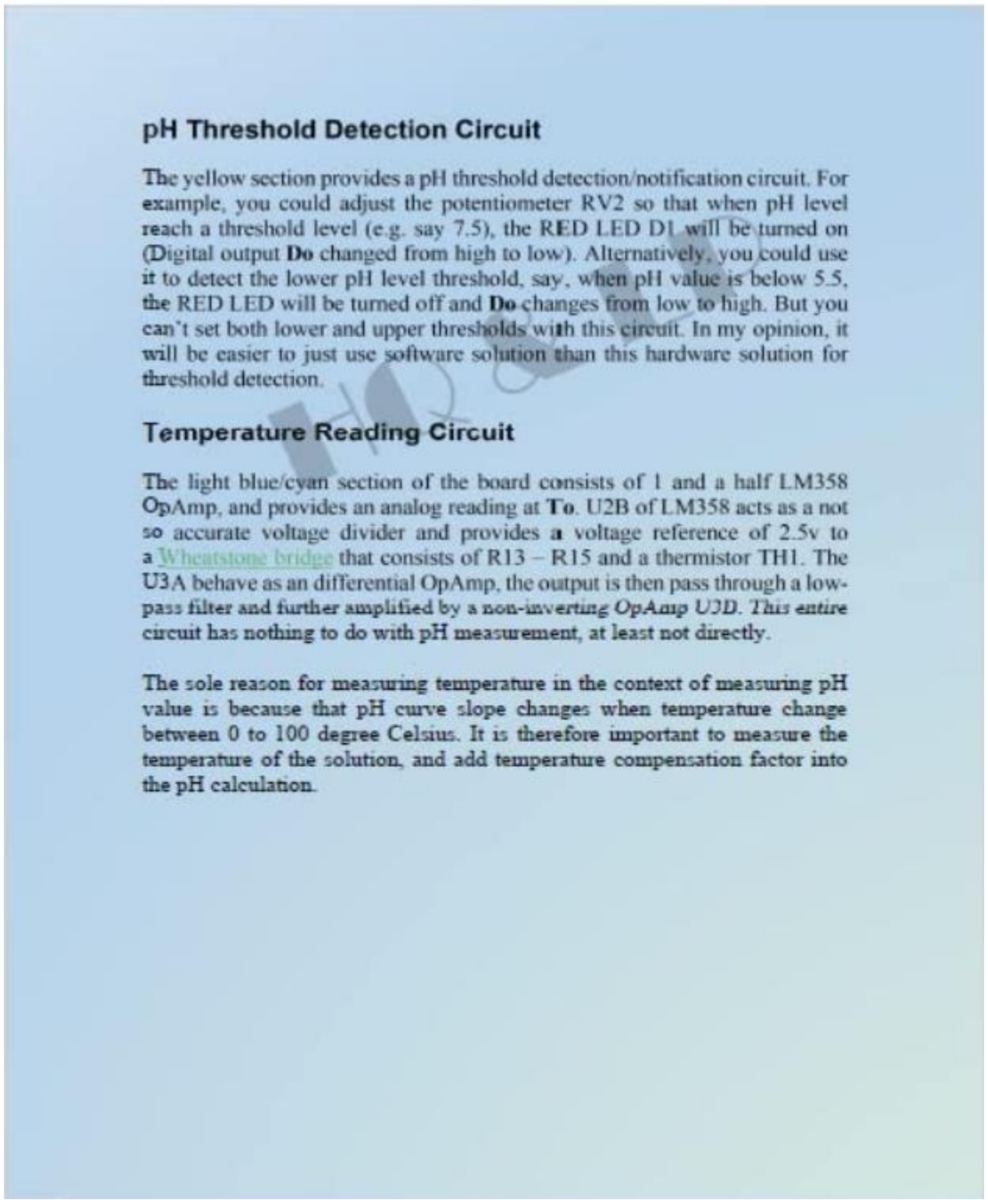
Hardware – The pH sensor board explained

The pH sensor board can be divided into 3 different sections based on its functionality.



pH Measurement Circuit

The light green section with the TLC4502 high-impedance operation amplifier basically consists of a voltage divider and a unity-gain amplifier. The pH output(Po) provided an analog output for pH measurement. As pH probe swing between positive and negative voltage, and since TLC4502 is operate with single power source, half of the TLC4502 is used as a voltage divider to provide a reference voltage of 2.5v to "float" the pH probe input so that the output of Po will be +/-2.5v based on pH value. A potentiometer RV1 is used for calibration purpose. This part of the circuit is all it needed for measuring the pH value



pH Threshold Detection Circuit

The yellow section provides a pH threshold detection/notification circuit. For example, you could adjust the potentiometer RV2 so that when pH level reach a threshold level (e.g. say 7.5), the RED LED D1 will be turned on (Digital output **Do** changed from high to low). Alternatively, you could use it to detect the lower pH level threshold, say, when pH value is below 5.5, the RED LED will be turned off and **Do** changes from low to high. But you can't set both lower and upper thresholds with this circuit. In my opinion, it will be easier to just use software solution than this hardware solution for threshold detection.

Temperature Reading Circuit

The light blue/cyan section of the board consists of 1 and a half LM358 OpAmp, and provides an analog reading at **To**. U2B of LM358 acts as a not so accurate voltage divider and provides a voltage reference of 2.5v to a Wheatstone bridge that consists of R13 – R15 and a thermistor TH1. The U3A behave as an differential OpAmp, the output is then pass through a low-pass filter and further amplified by a non-inverting OpAmp U3D. This entire circuit has nothing to do with pH measurement, at least not directly.

The sole reason for measuring temperature in the context of measuring pH value is because that pH curve slope changes when temperature change between 0 to 100 degree Celsius. It is therefore important to measure the temperature of the solution, and add temperature compensation factor into the pH calculation.

Pump

PULACO 160GPH Submersible Pump

Your Best Choice for Small Water Pump

1. The operating temperature range 40-95°F.

2. US Standard Plug is included in the package,which is only suitable for the US, Canada and Mexico.

3. If the water pump is stuck by anything like Mud, garbage, sea salt, etc., need to be cleaned with clean water before use.

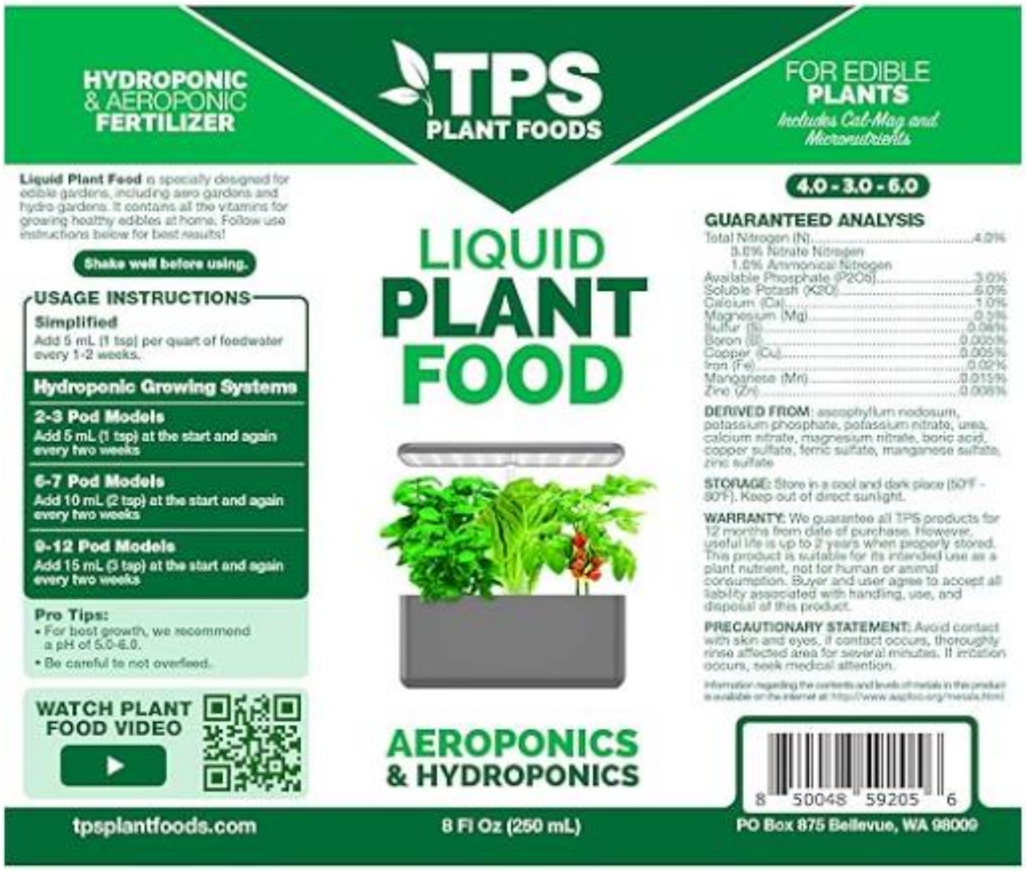
- Durable 10W super efficiency motor continually lift water column. Pond pump voltage is 110V 120V/60Hz.
- Max flow rate: 160GPH (600L/H).
- Max lift height: 4.9ft / 1.5M
- Long power cord: 6.0ft/1.8M
- nozzles: 0.5 inch , 0.5 inch
- 1 * 3.3ft Tubing (1/3")

PULACO: Your Creative Aquarium & Outdoor Garden Mate !

PULACO provides you with quality Aquarium & garden equipment and services. We are committed to being the best Outdoor gardening, hydroponic, Aquariums store you can trust.

Appendix E: Hydroponic Information

Liquid Nutrient Information



HYDROPONIC & AEROPONIC FERTILIZER

TPS PLANT FOODS

FOR EDIBLE PLANTS

Includes Cal-Mag and Micronutrients

Liquid Plant Food is specially designed for edible gardens, including aero gardens and hydro gardens. It contains all the vitamins for growing healthy edibles at home. Follow use instructions below for best results!

Shake well before using.

USAGE INSTRUCTIONS

Simplified
Add 5 mL (1 tsp) per quart of feedwater every 1-2 weeks.

Hydroponic Growing Systems

2-3 Pod Models
Add 5 mL (1 tsp) at the start and again every two weeks

6-7 Pod Models
Add 10 mL (2 tsp) at the start and again every two weeks

9-12 Pod Models
Add 15 mL (3 tsp) at the start and again every two weeks

Pro Tips:
- For best growth, we recommend a pH of 5.0-6.0.
- Be careful to not overfeed.

WATCH PLANT FOOD VIDEO

LIQUID PLANT FOOD

4.0 - 3.0 - 6.0

GUARANTEED ANALYSIS

Total Nitrogen (N)	4.0%
3.0% Nitrate Nitrogen	
1.0% Ammonical Nitrogen	
Available Phosphate (P2O5)	3.0%
Soluble Potash (K2O)	6.0%
Calcium (Ca)	1.0%
Magnesium (Mg)	0.5%
Sulfur (S)	0.08%
Boron (B)	0.005%
Copper (Cu)	0.005%
Iron (Fe)	0.02%
Manganese (Mn)	0.015%
Zinc (Zn)	0.008%

DERIVED FROM: ascophyllum nodosum, potassium phosphate, potassium nitrate, urea, calcium nitrate, magnesium nitrate, boric acid, copper sulfate, ferric sulfate, manganese sulfate, zinc sulfate

STORAGE: Store in a cool and dark place (50°F - 80°F). Keep out of direct sunlight.

WARRANTY: We guarantee all TPS products for 12 months from date of purchase. However, useful life is up to 2 years when properly stored. This product is suitable for its intended use as a plant nutrient, not for human or animal consumption. Buyer and user agree to accept all liability associated with handling, use, and disposal of this product.

PRECAUTIONARY STATEMENT: Avoid contact with skin and eyes. If contact occurs, thoroughly rinse affected area for several minutes. If irritation occurs, seek medical attention.

Information regarding the contents and levels of metals in this product is available on the internet at http://www.aapfco.org/metals.html

AEROPONICS & HYDROPONICS

8 50048 59205 6

tpsplantfoods.com

8 Fl Oz (250 mL)

PO Box 875 Bellevue, WA 98009

Microgreen Growing Information

Green Curled Ruffec Endive

Basic Info

Micro Greens Color:	Green leaves
Micro Greens Flavor:	mild, contrasting bitter

Growing Info

Seeding:	Approximately 1 Oz. of seed for a 10"x20" tray
Seed Presoak:	No
Growing Medium:	Hydroponic, soil
Germination Rate:	High
Germination Time:	2 to 3 days
Microgreens Harvest time:	8 to 15 days
Microgreens Ideal Harvest:	10 days
Baby Salad / Adult Stage Harvest:	16+ days

Other Info

Green Cureled Ruffec Endive Seeds:	Cichorium endivia
Preferred Medium:	Hydroponic for microgreens, soil for baby salad greens

Basil Microgreens

Basic Info

Latin Name:	*Ocimum basilcum*
Other Name(s):	
Microgreen Color:	green
Microgreen Flavor:	intense basil
Microgreen Texture:	soft
Nutrients:	Vitamins E, A, K, B6, and C, protein, calcium, iron, zinc, magnesium, copper, phosphorous, and potassium

Growing Info

Presoak:	No Soak
Preferred Growing Medium:	Hydroponic or Soil
Seeding Rate per 10"x 20" tray:	1 oz
Blackout Time:	4-7 days
Germination Time:	5-7 days
Estimated time to Harvest:	12-16 days

Mustard Seed Microgreens

Basic Info

Latin Name:	*Brassica rapa var japonica*
Nutrients:	Antioxidants, fiber, Vitamins A,C, E, and K
Microgreen Colors:	Green stems and green cotyledons with different shades
Microgreen Flavor:	Mustard spice with a citrus-like sweetness
Microgreen Texture:	Soft and slightly crunchy

Growing Info

Mustard Seed Presoak:	No
Microgreen Days to Maturity:	8-12
Growth Medium Preference:	Hydroponic or Soil

Other Info

Germ time:	2-3 days

Appendix F: Relevant Datasheets

ESP32

ESP32 Datasheet

Espressif Systems

October 8, 2016

About This Guide

This document provides introduction to the specifications of ESP32 hardware.
The document structure is as follows:

Chapter	Title	Subject
Chapter 1	Overview	An overview of ESP32, including featured solutions, basic and advanced features, applications and development support
Chapter 2	Pin Definitions	Introduction to the pin layout and descriptions
Chapter 3	Functional Description	Description of the major functional modules
Chapter 4	Peripheral Interface	Description of the peripheral interfaces integrated on ESP32
Chapter 5	Electrical Characteristics	The electrical characteristics and data of ESP32
Chapter 6	Package Information	The package details of ESP32
Chapter 7	Supported Resources	The related documents and community resources for ESP32
Appendix	Touch Sensor	The touch sensor design and layout guidelines

Release Notes

Date	Version	Release notes
2016.08	V1.0	First release

Disclaimer and Copyright Notice

Contents

List of Tables

List of Figures

1. Overview

ESP32 is a single chip 2.4 GHz Wi-Fi and Bluetooth combo chip designed with TSMC ultra low power 40 nm technology. It is designed and optimized for the best power performance, RF performance, robustness, versatility, features and reliability, for a wide variety of applications, and different power profiles.

1.1 Featured Solutions

1.1.1 Ultra Low Power Solution

ESP32 is designed for mobile, wearable electronics, and Internet of Things (IoT) applications. It has many features of the state-of-the-art low power chips, including fine resolution clock gating, power modes, and dynamic power scaling.

For instance, in a low-power IoT sensor hub application scenario, ESP32 is woken up periodically and only when a specified condition is detected; low duty cycle is used to minimize the amount of energy that the chip expends. The output power of the power amplifier is also adjustable to achieve an optimal trade off between communication range, data rate and power consumption.

Note:
For more information, refer to Section 3.7 RTC and Low-Power Management.

1.1.2 Complete Integration Solution

ESP32 is the most integrated solution for Wi-Fi + Bluetooth applications in the industry with less than 10 external components. ESP32 integrates the antenna switch, RF balun, power amplifier, low noise receive amplifier, filters, and power management modules. As such, the entire solution occupies minimal Printed Circuit Board (PCB) area.

ESP32 uses CMOS for single-chip fully-integrated radio and baseband, and also integrates advanced calibration circuitries that allow the solution to dynamically adjust itself to remove external circuit imperfections or adjust to changes in external conditions.

As such, the mass production of ESP32 solutions does not require expensive and specialized Wi-Fi test equipment.

1.2 Basic Protocols

1.2.1 Wi-Fi

- 802.11 b/g/n/e/i
- 802.11 n (2.4 GHz), up to 150 Mbps
- 802.11 e: QoS for wireless multimedia technology
- WMM-PS, UAPSD
- A-MPDU and A-MSDU aggregation
- Block ACK

- Fragmentation and defragmentation
- Automatic Beacon monitoring/scanning
- 802.11 i security features: pre-authentication and TSN
- Wi-Fi Protected Access (WPA)/WPA2/WPA2-Enterprise/Wi-Fi Protected Setup (WPS)
- Infrastructure BSS Station mode/SoftAP mode
- Wi-Fi Direct (P2P), P2P Discovery, P2P Group Owner mode and P2P Power Management
- UMA compliant and certified
- Antenna diversity and selection

Note:
For more information, refer to Section 3.5 Wi-Fi.

1.2.2 Bluetooth

- Compliant with Bluetooth v4.2 BR/EDR and BLE specification
- Class-1, class-2 and class-3 transmitter without external power amplifier
- Enhanced power control
- +10 dBm transmitting power
- NZIF receiver with -98 dBm sensitivity
- Adaptive Frequency Hopping (AFH)
- Standard HCI based on SDIO/SPI/UART
- High speed UART HCI, up to 4 Mbps
- BT 4.2 controller and host stack
- Service Discover Protocol (SDP)
- General Access Profile (GAP)
- Security Manage Protocol (SMP)
- Bluetooth Low Energy (BLE)
- ATT/GATT
- HID
- All GATT-based profile supported
- SPP-Like GATT-based profile
- BLE Beacon
- A2DP/AVRCP/SPP, HSP/HFP, RFCOMM
- CVSD and SBC for audio codec
- Bluetooth Piconet and Scatternet

1.3 MCU and Advanced Features

1.3.1 CPU and Memory

- Xtensa® Dual-Core 32-bit LX6 microprocessors, up to 600 DMIPS
- 448 KByte ROM
- 520 KByte SRAM
- 16 KByte SRAM in RTC
- QSPI Flash/SRAM, up to 4 x 16 MBytes
- Power supply: 2.2 V to 3.6 V

1.3.2 Clocks and Timers

- Internal 8 MHz oscillator with calibration
- Internal RC oscillator with calibration
- External 2 MHz to 40 MHz crystal oscillator
- External 32 kHz crystal oscillator for RTC with calibration
- Two timer groups, including 2 x 64-bit timers and 1 x main watchdog in each group
- RTC timer with sub-second accuracy
- RTC watchdog

1.3.3 Advanced Peripheral Interfaces

- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit D/A converters
- 10 × touch sensors
- Temperature sensor
- 4 × SPI
- 2 × I2S
- 2 × I2C
- 3 × UART
- 1 host (SD/eMMC/SDIO)
- 1 slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- CAN 2.0
- IR (TX/RX)
- Motor PWM
- LED PWM up to 16 channels
- Hall sensor
- Ultra low power analog pre-amplifier

1.3.4 Security

- IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI
- Secure boot
- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographic hardware acceleration:
  - AES
  - HASH (SHA-2) library
  - RSA
  - ECC
  - Random Number Generator (RNG)

1.3.5 Development Support

- SDK Firmware for fast on-line programming
- Open source toolchains based on GCC

Note:
For more information, refer to Chapter 7 Supported Resources.

1.4 Application

- Generic low power IoT sensor hub
- Generic low power IoT loggers
- Video streaming from camera
- Over The Top (OTT) devices
- Music players
  - Internet music players
  - Audio streaming devices
- Wi-Fi enabled toys
  - Loggers
  - Proximity sensing toys
- Wi-Fi enabled speech recognition devices
- Audio headsets
- Smart power plugs
- Home automation
- Mesh network

- Industrial wireless control
- Baby monitors
- Wearable electronics
- Wi-Fi location-aware devices
- Security ID tags
- Healthcare
  - Proximity and movement monitoring trigger devices
  - Temperature sensing loggers

1.5 Block Diagram

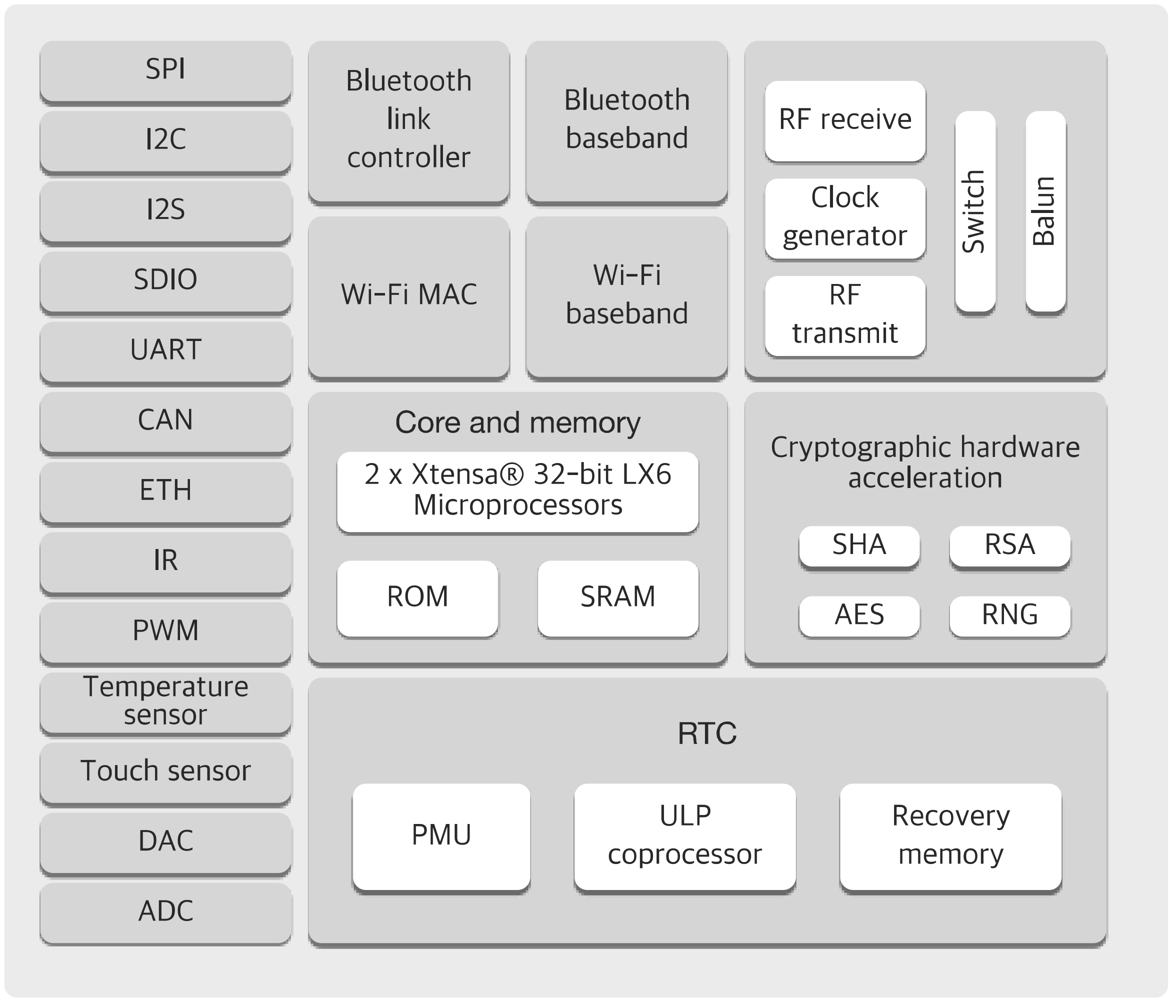


Figure 1: Function Block Diagram

2. Pin Definitions

2.1 Pin Layout

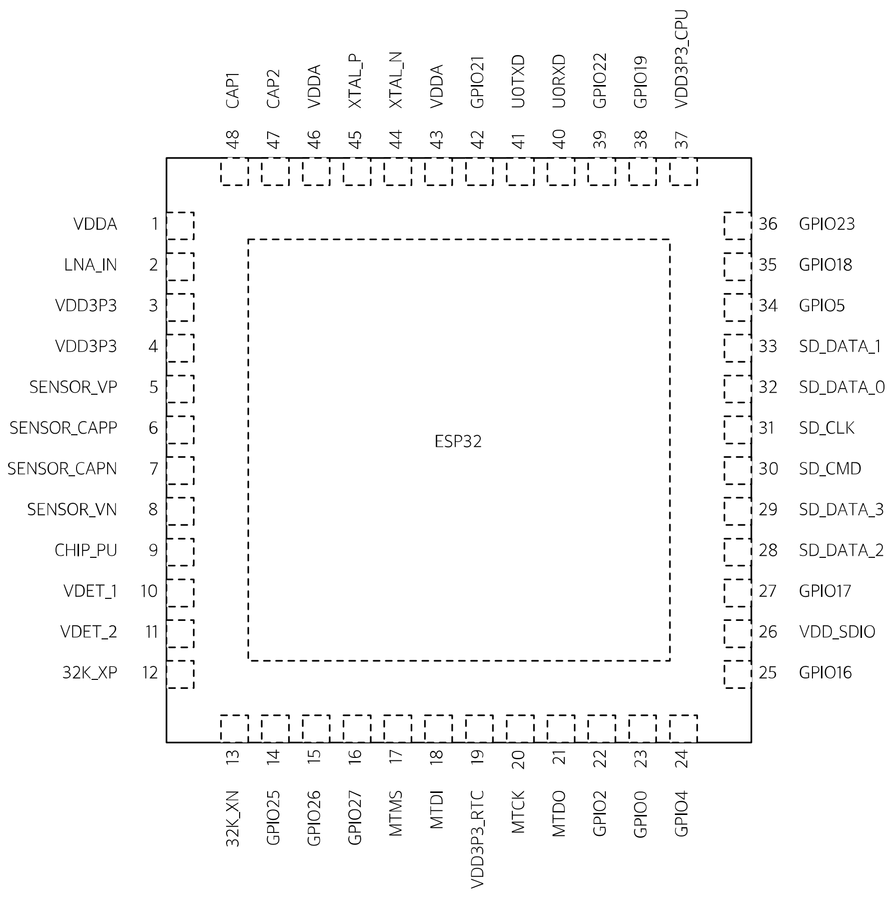


Figure 2: ESP32 Pin Layout

2.2 Pin Description

Table 1: Pin Description

Name	No.	Type	Function
			Analog
VDDA	1	P	Analog power supply (2.3V ~ 3.6V)
LNA_IN	2	I/O	RF input and output
VDD3P3	3	P	Amplifier power supply (2.3V ~ 3.6V)
VDD3P3	4	P	Amplifier power supply (2.3V ~ 3.6V)
			VDD3P3_RTC
SENSOR_VP	5	I	GPIO36, ADC_PRE_AMP, ADC1_CH0, RTC_GPIO0 Note: Connects 270 pF capacitor from SENSOR_VP to SENSOR_CAPP when used as ADC_PRE_AMP.

Name	No.	Type	Function
SENSOR_CAPP	6	I	GPIO37, ADC_PRE_AMP, ADC1_CH1, RTC_GPIO1 Note: Connects 270 pF capacitor from SENSOR_VP to SENSOR_CAPP when used as ADC_PRE_AMP.
SENSOR_CAPN	7	I	GPIO38, ADC1_CH2, ADC_PRE_AMP, RTC_GPIO2 Note: Connects 270 pF capacitor from SENSOR_VN to SENSOR_CAPN when used as ADC_PRE_AMP.
SENSOR_VN	8	I	GPIO39, ADC1_CH3, ADC_PRE_AMP, RTC_GPIO3 Note: Connects 270 pF capacitor from SENSOR_VN to SENSOR_CAPN when used as ADC_PRE_AMP.
CHIP_PU	9	I	Chip Enable (Active High) High: On, chip works properly Low: Off, chip works at the minimum power Note: Do not leave CHIP_PU pin floating
VDET_1	10	I	GPIO34, ADC1_CH6, RTC_GPIO4
VDET_2	11	I	GPIO35, ADC1_CH7, RTC_GPIO5
32K_XP	12	I/O	GPIO32, 32K_XP (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
32K_XN	13	I/O	GPIO33, 32K_XN (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8
GPIO25	14	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
GPIO26	15	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
GPIO27	16	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
MTMS	17	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
MTDI	18	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
VDD3P3_RTC	19	P	RTC IO power supply input (1.8V - 3.3V)
MTCK	20	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
MTDO	21	I/O	GPIO15, ADC2_CH3, TOUCH3, RTC_GPIO13, MTDO, HSPICS0, HS2_CMD, SD_CMD, EMAC_RXD3
GPIO2	22	I/O	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
GPIO0	23	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
GPIO4	24	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
VDD_SDIO			
GPIO16	25	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
VDD_SDIO	26	P	1.8V or 3.3V power supply output
GPIO17	27	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
SD_DATA_2	28	I/O	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
SD_DATA_3	29	I/O	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
SD_CMD	30	I/O	GPIO11, SD_CMD, SPICS0, HS1_CMD, U1RTS
SD_CLK	31	I/O	GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS

Name	No.	Type	Function
SD_DATA_0	32	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS
SD_DATA_1	33	I/O	GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS
VDD3P3_CPU			
GPIO5	34	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
GPIO18	35	I/O	GPIO18, VSPICLK, HS1_DATA7
GPIO23	36	I/O	GPIO23, VSPID, HS1_STROBE
VDD3P3_CPU	37	P	CPU IO power supply input (1.8V - 3.3V)
GPIO19	38	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
GPIO22	39	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
U0RXD	40	I/O	GPIO3, U0RXD, CLK_OUT2
U0TXD	41	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
GPIO21	42	I/O	GPIO21, VSPIHD, EMAC_TX_EN
Analog			
VDDA	43	I/O	Analog power supply (2.3V - 3.6V)
XTAL_N	44	O	External crystal output
XTAL_P	45	I	External crystal input
VDDA	46	P	Digital power supply for PLL (2.3V - 3.6V)
CAP2	47	I	Connects with a 3 nF capacitor and 20 kΩ resistor in parallel to CAP1
CAP1	48	I	Connects with a 10 nF series capacitor to ground

2.3 Power Scheme

ESP32 digital pins are divided into three different power domains:

- VDD3P3_RTC
- VDD3P3_CPU
- VDD_SDIO

VDD3P3_RTC is also the input power supply for RTC and CPU. **VDD3P3_CPU** is also the input power supply for CPU.
VDD_SDIO connects to the output of an internal LDO, whose input is **VDD3P3_RTC**. When **VDD_SDIO** is connected to the same PCB net together with **VDD3P3_RTC**; the internal LDO is disabled automatically.

The internal LDO can be configured as 1.8V, or the same voltage as **VDD3P3_RTC**. It can be powered off via software to minimize the current of Flash/SRAM during the Deep-sleep mode.

Note:
It is required that the power supply of **VDD3P3_RTC**, **VDD3P3_CPU** and analog must be stable before the pin **CHIP_PU** is set at high level.

2.4 Strapping Pins

ESP32 has 6 strapping pins:

- MTDI/GPIO12: internal pull-down
- GPIO0: internal pull-up
- GPIO2: internal pull-down
- GPIO4: internal pull-down
- MTDO/GPIO15: internal pull-up
- GPIO5: internal pull-up

Software can read the value of these 6 bits from the register "GPIO_STRAPPING".

During the chip power-on reset, the latches of the strapping pins sample the voltage level as strapping bits of "0" or "1", and hold these bits until the chip is powered down or shut down. The strapping bits configure the device boot mode, the operating voltage of VDD_SDIO and other system initial settings.

Each strapping pin is connected with its internal pull-up/pull-down during the chip reset. Consequently, if a strapping pin is unconnected or the connected external circuit is high-impendence, the internal weak pull-up/pull-down will determine the default input level of the strapping pins.

To change the strapping bit values, users can apply the external pull-down/pull-up resistances, or apply the host MCU's GPIOs to control the voltage level of these pins when powering on ESP32.

After reset, the strapping pins work as the normal functions pins.

Refer to Table 2 for detailed boot modes configuration by strapping pins.

Table 2: Strapping Pins

Voltage of Internal LDO (VDD_SDIO)					
Pin	Default	3.3V		1.8V	
MTDI	Pull-down	0		1	
Booting Mode					
Pin	Default	SPI Boot		Download Boot	
GPIO0	Pull-up	1		0	
GPIO2	Pull-down	Don't-care		0	
Debugging Log on U0TXD During Booting					
Pin	Default	U0TXD Toggling		U0TXD Silent	
MTDO	Pull-up	1		0	
Timing of SDIO Slave					
Pin	Default	Falling-edge Input Falling-edge Output	Falling-edge Input Rising-edge Output	Rising-edge Input Falling-edge Output	Rising-edge Input Rising-edge Output
MTDO	Pull-up	0	0	1	1
GPIO5	Pull-up	0	1	0	1

Note:
Firmware can configure register bits to change the setting of "Voltage of Internal LDO (VDD_SDIO)" and "Timing of SDIO Slave" after booting.

3. Functional Description

This chapter describes the functions implemented in ESP32.

3.1 CPU and Memory

3.1.1 CPU

ESP32 contains two low-power Xtensa® 32-bit LX6 microprocessors with the following features.

- 7-stage pipeline to support the clock frequency of up to 240 MHz
- 16/24-bit Instruction Set provides high code-density
- Support Floating Point Unit
- Support DSP instructions, such as 32-bit Multiplier, 32-bit Divider, and 40-bit MAC
- Support 32 interrupt vectors from about 70 interrupt sources

The dual CPUs interface through:

- Xtensa RAM/ROM Interface for instruction and data
- Xtensa Local Memory Interface for fast peripheral register access
- Interrupt with external and internal sources
- JTAG interface for debugging

3.1.2 Internal Memory

ESP32's internal memory includes:

- 448 KBytes ROM for booting and core functions
- 520 KBytes on-chip SRAM for data and instruction
- 8 KBytes SRAM in RTC, which is called RTC SLOW Memory and can be used for co-processor accessing during the Deep-sleep mode
- 8 KBytes SRAM in RTC, which is called RTC FAST Memory and can be used for data storage and main CPU during RTC Boot from the Deep-sleep mode
- 1 Kbit of EFUSE, of which 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including Flash-Encryption and Chip-ID

3.1.3 External Flash and SRAM

ESP32 supports 4 x 16 MBytes of external QSPI Flash and SRAM with hardware encryption based on AES to protect developer's programs and data.

ESP32 accesses external QSPI Flash and SRAM by the high-speed caches

- Up to 16 MBytes of external Flash are memory mapped into the CPU code space, supporting 8-bit, 16-bit and 32-bit access. Code execution is supported.

- Up to 8 MBytes of external Flash/SRAM are memory mapped into the CPU data space, supporting 8-bit, 16-bit and 32-bit access. Data read is supported on the Flash and SRAM. Data write is supported on the SRAM.

3.1.4 Memory Map

The structure of address mapping is shown in Figure 3. The memory and peripherals mapping of ESP32 is shown in Table 3.

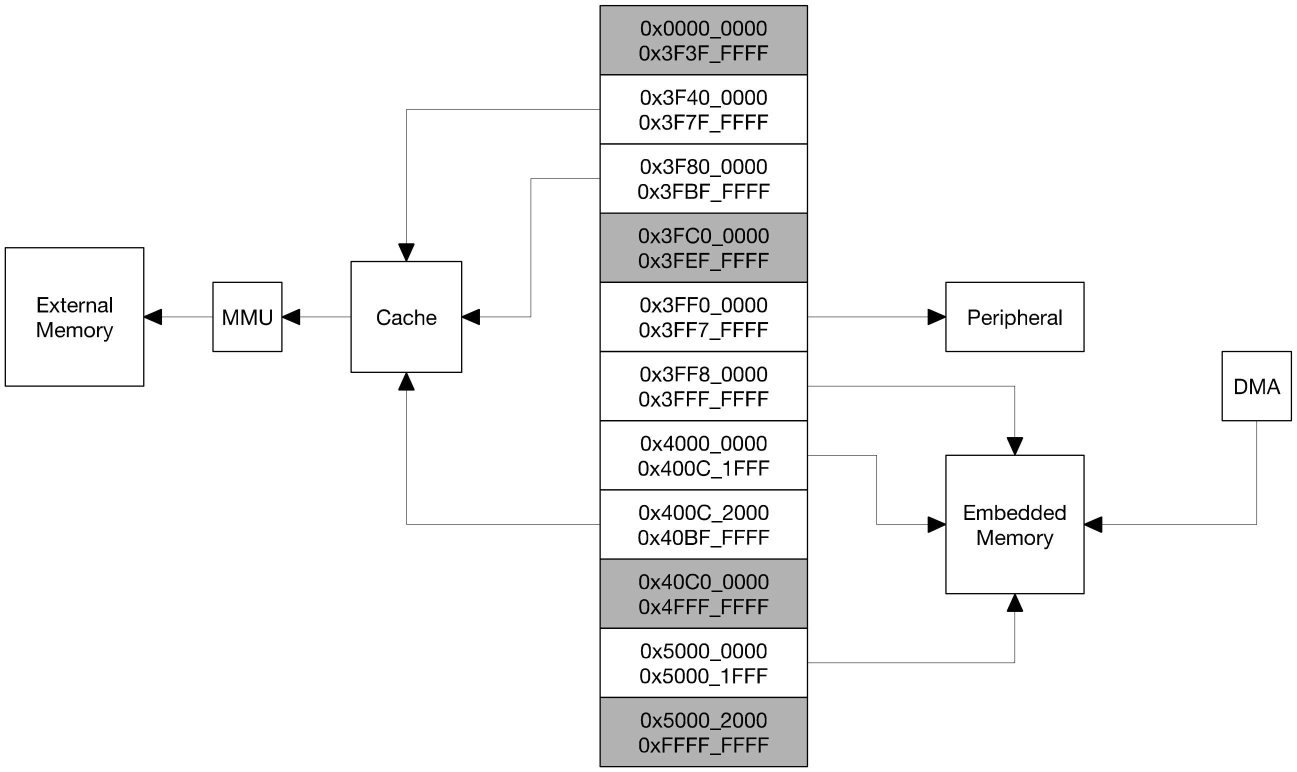


Figure 3: Address Mapping Structure

Table 3: Memory and Peripheral Mapping

Category	Target	Start Address	End Address	Size
Embedded Memory	Internal ROM 0	0x4000_0000	0x4005_FFFF	384 KB
	Internal ROM 1	0x3FF9_0000	0x3FF9_FFFF	64 KB
	Internal SRAM 0	0x4007_0000	0x4009_FFFF	192 KB
	Internal SRAM 1	0x3FFE_0000	0x3FFF_FFFF	128 KB
		0x400A_0000	0x400B_FFFF	
	Internal SRAM 2	0x3FFA_E000	0x3FFD_FFFF	200 KB
	RTC FAST Memory	0x3FF8_0000	0x3FF8_1FFF	8 KB
		0x400C_0000	0x400C_1FFF	
	RTC SLOW Memory	0x5000_0000	0x5000_1FFF	8 KB
External Memory	External Flash	0x3F40_0000	0x3F7F_FFFF	4 MB
		0x400C_2000	0x40BF_FFFF	11 MB
				248 KB
	External SRAM	0x3F80_0000	0x3FBF_FFFF	4 MB

Category	Target	Start Address	End Address	Size
Peripheral	DPort Register	0x3FF0_0000	0x3FF0_0FFF	4 KB
	AES Accelerator	0x3FF0_1000	0x3FF0_1FFF	4 KB
	RSA Accelerator	0x3FF0_2000	0x3FF0_2FFF	4 KB
	SHA Accelerator	0x3FF0_3000	0x3FF0_3FFF	4 KB
	Secure Boot	0x3FF0_4000	0x3FF0_4FFF	4 KB
	Cache MMU Table	0x3FF1_0000	0x3FF1_3FFF	16 KB
	PID Controller	0x3FF1_F000	0x3FF1_FFFF	4 KB
	UART0	0x3FF4_0000	0x3FF4_0FFF	4 KB
	SPI1	0x3FF4_2000	0x3FF4_2FFF	4 KB
	SPI0	0x3FF4_3000	0x3FF4_3FFF	4 KB
	GPIO	0x3FF4_4000	0x3FF4_4FFF	4 KB
	RTC	0x3FF4_8000	0x3FF4_8FFF	4 KB
	IO MUX	0x3FF4_9000	0x3FF4_9FFF	4 KB
	SDIO Slave	0x3FF4_B000	0x3FF4_BFFF	4 KB
	UDMA1	0x3FF4_C000	0x3FF4_CFFF	4 KB
	I2S0	0x3FF4_F000	0x3FF4_FFFF	4 KB
	UART1	0x3FF5_0000	0x3FF5_0FFF	4 KB
	I2C0	0x3FF5_3000	0x3FF5_3FFF	4 KB
	UDMA0	0x3FF5_4000	0x3FF5_4FFF	4 KB
	SDIO Slave	0x3FF5_5000	0x3FF5_5FFF	4 KB
	RMT	0x3FF5_6000	0x3FF5_6FFF	4 KB
	PCNT	0x3FF5_7000	0x3FF5_7FFF	4 KB
	SDIO Slave	0x3FF5_8000	0x3FF5_8FFF	4 KB
	LED PWM	0x3FF5_9000	0x3FF5_9FFF	4 KB
	Efuse Controller	0x3FF5_A000	0x3FF5_AFFF	4 KB
	Flash Encryption	0x3FF5_B000	0x3FF5_BFFF	4 KB
	PWM0	0x3FF5_E000	0x3FF5_EFFF	4 KB
	TIMG0	0x3FF5_F000	0x3FF5_FFFF	4 KB
	TIMG1	0x3FF6_0000	0x3FF6_0FFF	4 KB
	SPI2	0x3FF6_4000	0x3FF6_4FFF	4 KB
	SPI3	0x3FF6_5000	0x3FF6_5FFF	4 KB
	SYSCON	0x3FF6_6000	0x3FF6_6FFF	4 KB
	I2C1	0x3FF6_7000	0x3FF6_7FFF	4 KB
	SDMMC	0x3FF6_8000	0x3FF6_8FFF	4 KB
	EMAC	0x3FF6_9000	0x3FF6_AFFF	8 KB
	PWM1	0x3FF6_C000	0x3FF6_CFFF	4 KB
	I2S1	0x3FF6_D000	0x3FF6_DFFF	4 KB
	UART2	0x3FF6_E000	0x3FF6_EFFF	4 KB
	PWM2	0x3FF6_F000	0x3FF6_FFFF	4 KB
	PWM3	0x3FF7_0000	0x3FF7_0FFF	4 KB
	RNG	0x3FF7_5000	0x3FF7_5FFF	4 KB

3.2 Timers and Watchdogs

3.2.1 64-bit Timers

There are four general-purpose timers embedded in the ESP32. They are all 64-bit generic timers which are based on 16-bit prescalers and 64-bit auto-reload-capable up/downcounters.

The timers feature:

- A 16-bit clock prescaler, from 2 to 65536
- A 64-bit time-base counter
- Configurable up/down time-base counter: incrementing or decrmenting
- Halt and resume of time-base counter
- Auto-reload at alarming
- Software-controlled instant reload
- Level and edge interrupt generation

3.2.2 Watchdog Timers

The ESP32 has three watchdog timers: one in each of the two timer modules (called the Main Watchdog Timer, or MWDT) and one in the RTC module (called the RTC Watchdog Timer, or RWDT). These watchdog timers are intended to recover from an unforeseen fault, causing the application program to abandon its normal sequence. A watchdog timer has 4 stages. Each stage may take one of three or four actions on expiry of a programmed time period for this stage unless the watchdog is fed or disabled. The actions are: interrupt, CPU reset, and core reset, and system reset. Only the RWDT can trigger the system reset, and is able to reset the entire chip, including the RTC itself. A timeout value can be set for each stage individually.

During Flash boot the RWDT and the first MWDT start automatically in order to detect and recover from booting problems.

The ESP32 watchdogs have the following features:

- 4 stages, each can be configured or disabled separately
- Programmable time period for each stage
- One of 3 or 4 possible actions (interrupt, CPU reset, core reset, and system reset) on expiration of each stage
- 32-bit expiry counter
- Write protection, to prevent the RWDT and MWDT configuration from being inadvertently altered
- SPI Flash boot protection
  If the boot process from an SPI Flash does not complete within a predetermined time period, the watchdog will reboot the entire system.

3.3 System Clocks

3.3.1 CPU Clock

Upon reset, an external crystal clock source (2 MHz ~ 60 MHz), is selected as the default CPU clock. The external crystal clock source also connects to a PLL to generate a high frequency clock (typically 160 MHz).

In addition to this, ESP32 has an internal 8 MHz oscillator, of which the accuracy is guaranteed by design and is stable over temperature (within 1% accuracy). Hence, the application can then select from the external crystal clock source, the PLL clock or the internal 8 MHz oscillator. The selected clock source drives the CPU clock, directly or after division, depending on the application.

3.3.2 RTC Clock

The RTC clock has five possible sources:

- external low speed (32 kHz) crystal clock
- external crystal clock divided by 4
- internal RC oscillator (typically about 150 kHz and adjustable)
- internal 8 MHz oscillator
- internal 31.25 kHz clock (derived from the internal 8 MHz oscillator divided by 256)

When the chip is in the normal power mode and needs faster CPU accessing, the application can choose the external high speed crystal clock divided by 4 or the internal 8 MHz oscillator. When the chip operates in the low power mode, the application chooses the external low speed (32 kHz) crystal clock, the internal RC clock or the internal 31.25 kHz clock.

3.3.3 Audio PLL Clock

The audio clock is generated by the ultra low noise fractional-N PLL. The output frequency of the audio PLL is programmable, from 16 MHz to 128 MHz, given by the following formula:

$$f_{\text{out}} = \frac{f_{\text{xtal}} N_{\text{div}}}{M_{\text{div}} 2^{K_{\text{div}}}}$$

where f_{out} is the output frequency, f_{xtal} is the frequency of the crystal oscillator, and N_{div}, M_{div} and K_{div} are all integer values, configurable by registers.

3.4 Radio

The ESP32 radio consists of the following main blocks:

- 2.4 GHz receiver
- 2.4 GHz transmitter
- bias and regulators
- balun and transmit-receive switch
- clock generator

3.4.1 2.4 GHz Receiver

The 2.4 GHz receiver down-converts the 2.4 GHz RF signal to quadrature baseband signals and converts them to the digital domain with 2 high-resolution, high-speed ADCs. To adapt to varying signal channel conditions, RF filters, Automatic Gain Control (AGC), DC offset cancelation circuits and baseband filters are integrated within ESP32.

3.4.2 2.4 GHz Transmitter

The 2.4 GHz transmitter up-converts the quadrature baseband signals to the 2.4 GHz RF signal, and drives the antenna with a high powered Complementary Metal Oxide Semiconductor (CMOS) power amplifier. The use of digital calibration further improves the linearity of the power amplifier, enabling state-of-the-art performance of delivering +20.5 dBm of average power for 802.11b transmission and +17 dBm for 802.11n transmission.
Additional calibrations are integrated to cancel any imperfections of the radio, such as:

- Carrier leakage
- I/Q phase matching
- Baseband nonlinearities
- RF nonlinearities
- Antenna matching

These built-in calibration routines reduce the amount of time and required for product test and make test equipment unnecessary.

3.4.3 Clock Generator

The clock generator generates quadrature 2.4 GHz clock signals for the receiver and transmitter. All components of the clock generator are integrated on the chip, including all inductors, varactors, filters, regulators and dividers. The clock generator has built-in calibration and self test circuits. Quadrature clock phases and phase noise are optimized on-chip with patented calibration algorithms to ensure the best performance of the receiver and transmitter.

3.5 Wi-Fi

ESP32 implements TCP/IP, full 802.11 b/g/n/e/i WLAN MAC protocol, and Wi-Fi Direct specification. It supports Basic Service Set (BSS) STA and SoftAP operations under the Distributed Control Function (DCF) and P2P group operation compliant with the latest Wi-Fi P2P protocol.

Passive or active scanning, as well as the P2P discovery procedure are performed autonomously when initiated by appropriate commands. Power management is handled with minimum host interaction to minimize active duty period.

3.5.1 Wi-Fi Radio and Baseband

The ESP32 Wi-Fi Radio and Baseband support the following features:

- 802.11b and 802.11g data-rates
- 802.11n MCS0-7 in both 20 MHz and 40 MHz bandwidth
- 802.11n MCS32
- 802.11n 0.4 μS guard-interval
- Data-rate up to 150 Mbps
- Receiving STBC 2x1
- Up to 21 dBm transmitting power
- Adjustable transmitting power

- Antenna diversity and selection (software-managed hardware)

3.5.2 Wi-Fi MAC

The ESP32 Wi-Fi MAC applies low level protocol functions automatically as follows:

- Request To Send (RTS), Clear To Send (CTS) and Acknowledgement (ACK/BA)
- Fragmentation and defragmentation
- Aggregation AMPDU and AMSDU
- WMM, U-APSD
- 802.11 e: QoS for wireless multimedia technology
- CCMP (CBC-MAC, counter mode), TKIP (MIC, RC4), WAPI (SMS4), WEP (RC4) and CRC
- Frame encapsulation (802.11h/RFC 1042)
- Automatic beacon monitoring/scanning

3.5.3 Wi-Fi Firmware

The ESP32 Wi-Fi Firmware provides the following functions:

- Infrastructure BSS Station mode / P2P mode / softAP mode support
- P2P Discovery, P2P Group Owner, P2P Group Client and P2P Power Management
- WPA/WPA2-Enterprise and WPS driver
- Additional 802.11i security features such as pre-authentication and TSN
- Open interface for various upper layer authentication schemes over EAP such as TLS, PEAP, LEAP, SIM, AKA or customer specific
- Clock/power gating combined with 802.11-compliant power management dynamically adapted to current connection condition providing minimal power consumption
- Adaptive rate fallback algorithm sets the optimal transmission rate and transmit power based on actual Signal Noise Ratio (SNR) and packet loss information
- Automatic retransmission and response on MAC to avoid packet discarding on slow host environment

3.5.4 Packet Traffic Arbitration (PTA)

ESP32 has a configurable Packet Traffic Arbitration (PTA) that provides flexible and exact timing Bluetooth co-existence support. It is a combination of both Frequency Division Multiplexing (FDM) and Time Division Multiplexing (TDM), and coordinates the protocol stacks.

- It is preferable that Wi-Fi works in the 20 MHz bandwidth mode to decrease its interference with BT.
- BT applies AFH (Adaptive Frequency Hopping) to avoid using the channels within Wi-Fi bandwidth.
- Wi-Fi MAC limits the time duration of Wi-Fi packets, and does not transmit the long Wi-Fi packets by the lowest data-rates.
- Normally BT packets are of higher priority than normal Wi-Fi packets.
- Protect the critical Wi-Fi packets, including beacon transmission and receiving, ACK/BA transmission and receiving.

- Protect the highest BT packets, including inquiry response, page response, LMP data and response, park beacons, the last poll period, SCO/eSCO slots, and BLE event sequence.
- Wi-Fi MAC applies CTS-to-self packet to protect the time duration of BT transfer.
- In the P2P Group Own (GO) mode, Wi-Fi MAC applies a Notice of Absence (NoA) packet to disable Wi-Fi transfer to reserve time for BT.
- In the STA mode, Wi-Fi MAC applies a NULL packet with the Power-Save bit to disable WiFi transfer to reserve time for BT.

3.6 Bluetooth

ESP32 integrates Bluetooth link controller and Bluetooth baseband, which carry out the baseband protocols and other low-level link routines, such as modulation/demodulation, packets processing, bit stream processing, frequency hopping, etc.

3.6.1 Bluetooth Radio and Baseband

The ESP32 Bluetooth Radio and Baseband support the following features:

- Class-1, class-2 and class-3 transmit output powers and over 30 dB dynamic control range
- π/4 DQPSK and 8 DPSK modulation
- High performance in NZIF receiver sensitivity with over 98 dB dynamic range
- Class-1 operation without external PA
- Internal SRAM allows full speed data transfer, mixed voice and data, and full piconet operation
- Logic for forward error correction, header error control, access code correlation, CRC, demodulation, encryption bit stream generation, whitening and transmit pulse shaping
- ACL, SCO, eSCO and AFH
- A-law, μ-law and CVSD digital audio CODEC in PCM interface
- SBC audio CODEC
- Power management for low power applications
- SMP with 128-bit AES

3.6.2 Bluetooth Interface

- Provides UART HCI interface, up to 4 Mbps
- Provides SDIO / SPI HCI interface
- Provides I2C interface for the host to do configuration
- Provides PCM / I2S audio interface

3.6.3 Bluetooth Stack

The Bluetooth stack of ESP32 is compliant with Bluetooth v4.2 BR / EDR and BLE specification.

3.6.4 Bluetooth Link Controller

The link controller operates in three major states: standby, connection and sniff. It enables multi connection and other operations like inquiry, page, and secure simple pairing, and therefore enables Piconet and Scatternet. Below are the features:

- Classic Bluetooth
  - Device Discovery (inquiry and inquiry scan)
  - Connection establishment (page and page scan)
  - Multi connections
  - Asynchronous data reception and transmission
  - Synchronous links (SCO/eSCO)
  - Master/Slave Switch
  - Adaptive Frequency Hopping and Channel assessment
  - Broadcast encryption
  - Authentication and encryption
  - Secure Simple Pairing
  - Multi-point and scatternet management
  - Sniff mode
  - Connectionless Slave Broadcast (transmitter and receiver)
  - Enhanced power control
  - Ping
- Bluetooth Low Energy
  - Advertising
  - Scanning
  - Multiple connections
  - Asynchronous data reception and transmission
  - Adaptive Frequency Hopping and Channel assessment
  - Connection parameter update
  - Date Length Extension
  - Link Layer Encryption
  - LE Ping

3.7 RTC and Low-Power Management

With the advanced power management technologies, ESP32 can switch between different power modes (see Table 4).

- Power mode
  - Active mode: The chip radio is powered on. The chip can receive, transmit, or listen.
  - Modem-sleep mode: The CPU is operational and the clock is configurable. The Wi-Fi/Bluetooth baseband and radio are disabled.
  - Light-sleep mode: The CPU is paused. The RTC and ULP-coprocessor are running. Any wake-up events (MAC, host, RTC timer, or external interrupts) will wake up the chip.
  - Deep-sleep mode: Only RTC is powered on. Wi-Fi and Bluetooth connection data are stored in RTC memory. The ULP-coprocessor can work.
  - Hibernation mode: The internal 8MHz oscillator and ULP-coprocessor are disabled. The RTC recovery memory are power-down. Only one RTC timer on the slow clock and some RTC GPIOs are active. The RTC timer or the RTC GPIOs can wake up the chip from the Hibernation mode.
- Sleep Pattern
  - Association sleep pattern: The power mode switches between the active mode and Modem-sleep/Light-sleep mode during this sleep pattern. The CPU, Wi-Fi, Bluetooth, and radio are woken up at predetermined intervals to keep Wi-Fi/BT connections alive.
  - ULP sensor-monitored pattern: The main CPU is in the Deep-sleep mode. The ULP co-processor does sensor measurements and wakes up the main system, based on the measured data from sensors.

Table 4: Functionalities Depending on the Power Modes

Power mode	Active	Modem-sleep	Light-sleep	Deep-sleep	Hibernation
Sleep pattern	Association sleep pattern			ULP sensor-monitored pattern	-
CPU	ON	PAUSE	ON	OFF	OFF
Wi-Fi/BT baseband and radio	ON	OFF	OFF	OFF	OFF
RTC	ON	ON	ON	ON	OFF
ULP co-processor	ON	ON	ON	ON/OFF	OFF

The power consumption varies with different power modes/sleep patterns and work status of functional modules (see Table 5).

Table 5: Power Consumption by Power Modes

Power mode	Description	Power consumption
Active (RF working)	Wi-Fi Tx packet 13 dBm ~ 21 dBm	160 ~ 260 mA
	Wi-Fi / BT Tx packet 0 dBm	120 mA
	Wi-Fi / BT Rx and listening	80 ~ 90 mA
	Association sleep pattern (by Light-sleep)	0.9 mA@DTIM3, 1.2 mA@DTIM1
Modem-sleep	The CPU is powered on.	Max speed: 20 mA
		Normal speed: 5 ~ 10 mA
		Slow speed: 3 mA
Light-sleep	-	0.8 mA
Deep-sleep	The ULP co-processor is powered on.	0.15 mA
	ULP sensor-monitored pattern	25 μA @1% duty
	RTC timer + RTC memory	10 μA
Hibernation	RTC timer only	2.5 μA

Note:
For more information about RF power consumption, refer to Section 5.3 RF Power Consumption Specifications.

4. Peripheral Interface

4.1 General Purpose Input / Output Interface (GPIO)

ESP32 has 48 GPIO pins which can be assigned to various functions by programming the appropriate registers. There are several kinds of GPIOs: digital only GPIOs, analog enabled GPIOs, capacitive touch enabled GPIOs, etc. Analog enabled GPIOs can be configured as digital GPIOs. Capacitive touch enabled GPIOs can be configured as digital GPIOs.

Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, the input value can be read through the register. The input can also be set to edge-trigger or level-trigger to generate CPU interrupts. In short, the digital IO pins are bi-directional, non-inverting and tristate, including input and output buffer with tristate control. These pins can be multiplexed with other functions, such as the SDIO interface, UART, SI, etc. For low power operations, the GPIOs can be set to hold their states.

4.2 Analog-to-Digital Converter (ADC)

ESP32 integrates 12-bit SAR ADCs and supports measurements on 18 channels (analog enabled pins). Some of these pins can be used to build a programmable gain amplifier which is used for the measurement of small analog signals. The ULP-coprocessor in ESP32 is also designed to measure the voltages while operating in the sleep mode, to enable low power consumption; the CPU can be woken up by a threshold setting and/or via other triggers.

With the appropriate setting, the ADCs and the amplifier can be configured to measure voltages for a maximum of 18 pins.

4.3 Ultra Low Noise Analog Pre-Amplifier

ESP32 integrates an ultra low noise analog pre-amplifier that outputs to the ADC. The amplification ratio is given by the size of a pair of sampling capacitors that are placed off-chip. By using a larger capacitor, the sampling noise is reduced, but the settling time will be increased. The amplification ratio is also limited by the amplifier which peaks at about 60 dB gain.

4.4 Hall Sensor

ESP32 integrates a Hall sensor based on an N-carrier resistor. When the chip is in the magnetic field, the Hall sensor develops a small voltage laterally on the resistor, which can be directly measured by the ADC, or amplified by the ultra low noise analog pre-amplifier and then measured by the ADC.

4.5 Digital-to-Analog Converter (DAC)

Two 8-bit DAC channels can be used to convert two digital signals into two analog voltage signal outputs. The design structure is composed of integrated resistor strings and a buffer. This dual DAC supports power supply as input voltage reference and can drive other circuits. The dual channels support independent conversions.

4.6 Temperature Sensor

The temperature sensor generates a voltage that varies with temperature. The voltage is internally converted via an analog-to-digital converter into a digital code.

The temperature sensor has a range of -40°C to 125°C. As the offset of the temperature sensor varies from chip to chip due to process variation, together with the heat generated by the Wi-Fi circuitry itself (which affects measurements), the internal temperature sensor is only suitable for applications that detect temperature changes instead of absolute temperatures and for calibration purposes as well.

However, if the user calibrates the temperature sensor and uses the device in a minimally powered-on application, the results could be accurate enough.

4.7 Touch Sensor

ESP32 offers 10 capacitive sensing GPIOs which detect capacitive variations introduced by the GPIO's direct contact or close proximity with a finger or other objects. The low noise nature of the design and high sensitivity of the circuit allow relatively small pads to be used. Arrays of pads can also be used so that a larger area or more points can be detected. The 10 capacitive sensing GPIOs are listed in Table 6.

Table 6: Capacitive Sensing GPIOs Available on ESP32

Capacitive sensing signal name	Pin name
T0	GPIO4
T1	GPIO0
T2	GPIO2
T3	MTDO
T4	MTCK
T5	MTD1
T6	MTMS
T7	GPIO27
T8	32K_XN
T9	32K_XP

Note:
For more information about the touch sensor design and layout, refer to Appendix A Touch Sensor.

4.8 Ultra-Lower-Power Coprocessor

The ULP processor and RTC memory remains powered on during the Deep-sleep mode. Hence, the developer can store a program for the ULP processor in the RTC memory to access the peripheral devices, internal timers and internal sensors during the Deep-sleep mode. This is useful for designing applications where the CPU needs to be woken up by an external event, or timer, or a combination of these events, while maintaining minimal power consumption.

4.9 Ethernet MAC Interface

An IEEE-802.3-2008-compliant Media Access Controller (MAC) is provided for Ethernet LAN communications. ESP32 requires an external physical interface device (PHY) to connect to the physical LAN bus (twisted-pair, fiber, etc.). The PHY is connected to ESP32 through 17 signals of MII or 9 signals of RMII. With the Ethernet MAC (EMAC) interface, the following features are supported:

- 10 Mbps and 100 Mbps rates
- Dedicated DMA controller allowing high-speed transfer between the dedicated SRAM and Ethernet MAC
- Tagged MAC frame (VLAN support)
- Half-duplex (CSMA/CD) and full-duplex operation
- MAC control sublayer (control frames)
- 32-bit CRC generation and removal
- Several address filtering modes for physical and multicast address (multicast and group addresses)
- 32-bit status code for each transmitted or received frame
- Internal FIFOs to buffer transmit and receive frames. The transmit FIFO and the receive FIFO are both 512 words (32-bit)
- Hardware PTP (precision time protocol) in accordance with IEEE 1588 2008 (PTP V2)
- 25 MHz/50 MHz clock output

4.10 SD/SDIO/MMC Host Controller

An SD/SDIO/MMC host controller is available on ESP32 which supports the following features:

- Secure Digital memory (SD mem Version 3.0 and Version 3.01)
- Secure Digital I/O (SDIO Version 3.0)
- Consumer Electronics Advanced Transport Architecture (CE-ATA Version 1.1)
- Multimedia Cards (MMC Version 4.41, eMMC Version 4.5 and Version 4.51)

The controller allows clock output at up to 80 MHz and in three different data-bus modes: 1-bit, 4-bit and 8-bit. It supports two SD/SDIO/MMC4.41 cards in 4-bit data-bus mode. It also supports one SD card operating at 1.8 V level.

4.11 Universal Asynchronous Receiver Transmitter (UART)

ESP32 has three UART interfaces, i.e. UART0, UART1 and UART2, which provide asynchronous communication (RS232 and RS485) and IrDA support, and communicate at up to 5 Mbps. UART provides hardware management of the CTS and RTS signals and software flow control (XON and XOFF). All of the interfaces can be accessed by the DMA controller or directly by CPU.

4.12 I2C Interface

ESP32 has two I2C bus interfaces which can serve as I2C master or slave depending on the user's configuration. The I2C interfaces support:

- Standard mode (100 kbit/s)
- Fast mode (400 kbit/s)
- Up to 5 MHz, but constrained by SDA pull up strength
- 7-bit/10-bit addressing mode
- Dual addressing mode

Users can program command registers to control I2C interfaces to have more flexibility.

4.13 I2S Interface

Two standard I2S interfaces are available in ESP32. They can be operated in the master or slave mode, in full duplex and half-duplex communication modes, and can be configured to operate with an 8-/16-/32-/40-/48-bit resolution as input or output channels. BCK clock frequency from 10 kHz up to 40 MHz are supported. When one or both of the I2S interfaces are configured in the master mode, the master clock can be output to the external DAC/CODEC.

Both of the I2S interfaces have dedicated DMA controllers. PDM and BT PCM interfaces are supported.

4.14 Infrared Remote Controller

The infrared remote controller supports eight channels of infrared remote transmission and receiving. Through programming the pulse waveform, it supports various infrared protocols. Eight channels share a 512 x 32-bit block of memory to store the transmitting or receiving waveform.

4.15 Pulse Counter

The pulse counter captures pulse and counts pulse edges through seven modes. It has 8 channels; each channel captures four signals at a time. The four input signals include two pulse signals and two control signals. When the counter reaches a defined threshold, an interrupt is generated.

4.16 Pulse Width Modulation (PWM)

The Pulse Width Modulation (PWM) controller can be used for driving digital motors and smart lights. The controller consists of PWM timers, the PWM operator and a dedicated capture sub-module. Each timer provides timing in synchronus or independent form, and each PWM operator generates the waveform for one PWM channel. The dedicated capture sub-module can accurately capture external timing events.

4.17 LED PWM

The LED PWM controller can generate 16 independent channels of digital waveforms with the configurable periods and configurable duties.

The 16 channels of digital waveforms operate at 80 MHz APB clock, among which 8 channels have the option of using the 8 MHz oscillator clock. Each channel can select a 20-bit timer with configurable counting range and its accuracy of duty can be up to 16 bits with the 1 ms period.

The software can change the duty immediately. Moreover, each channel supports step-by-step duty increasing or decreasing automatically. It is useful for the LED RGB color gradient generator.

4.18 Serial Peripheral Interface (SPI)

ESP32 features three SPIs (SPI, HSPI and VSPI) in slave and master modes in 1-line full-duplex and 1/2/4-line half-duplex communication modes. These SPIs also support the following general-purpose SPI features:

- 4 timing modes of the SPI format transfer that depend on the polarity (POL) and the phase (PHA)
- up to 80 MHz and the divided clocks of 80 MHz
- up to 64-Byte FIFO

All SPIs can also be used to connect to the external Flash/SRAM and LCD. Each SPI can be served by DMA controllers.

4.19 Accelerator

ESP32 is equipped with hardware accelerators of general algorithms, such as AES (FIPS PUB 197), SHA (FIPS PUB 180-4), RSA, and ECC, which support independent arithmetic such as Big Integer Multiplication and Big Integer Modular Multiplication. The maximum operation length for RSA, ECC, Big Integer Multiply and Big Integer Modular Multiplication is 4096 bits.

The hardware accelerators greatly improve operation speed and reduce software complexity. They also support code encryption and dynamic decryption which ensures that codes in the Flash will not be stolen.

5. Electrical Characteristics

Note:

The specifications in this charpter are tested in general condition: V_{BAT} = 3.3V, T_A = 27°C, unless otherwise specified.

5.1 Absolute Maximum Ratings

Table 7: Absolute Maximum Ratings

Parameter	Symbol	Min	Max	Unit
Input low voltage	V_{IL}	-0.3	0.25×V_{IO}	V
Input high voltage	V_{IH}	0.75×V_{IO}	3.3	V
Input leakage current	I_{IL}	-	50	nA
Output low voltage	V_{OL}	-	0.1×V_{IO}	V
Output high voltage	V_{OH}	0.8×V_{IO}	-	V
Input pin capacitance	C_{pad}	-	2	pF
VDDIO	V_{IO}	1.8	3.3	V
Maximum drive capability	I_{MAX}	-	12	mA
Storage temperature range	T_{STR}	-40	150	°C

5.2 Recommended Operating Conditions

Table 8: Recommended Operating Conditions

Parameter	Symbol	Min	Typ	Max	Unit
Battery regulator supply voltage	V_{BAT}	2.8	3.3	3.6	V
I/O supply voltage	V_{IO}	1.8	3.3	3.6	V
Operating temperature range	T_{OPR}	-40	-	125	°C
CMOS low level input voltage	V_{IL}	0	-	0.3 x V_{IO}	V
CMOS high level input voltage	V_{IH}	0.7 x V_{IO}	-	V_{IO}	V
CMOS threshold voltage	V_{TH}	-	0.5 x V_{IO}	-	V

5.3 RF Power Consumption Specifications

The current consumption measurements are conducted with 3.0 V supply and 25°C ambient, at antenna port. All the transmitters' measurements are based on 90% duty cycle and continuous transmit mode.

Table 9: RF Power Consumption Specifications

Mode	Min	Typ	Max	Unit
Transmit 802.11b, DSSS 1 Mbps, POUT = +19.5 dBm	-	225	-	mA
Transmit 802.11b, CCK 11 Mbps, POUT = +18.5 dBm	-	205	-	mA
Transmit 802.11g, OFDM 54 Mbps, POUT = +16 dBm	-	160	-	mA
Transmit 802.11n, MCS7, POUT = +14 dBm	-	152	-	mA
Receive 802.11b, packet length = 1024 bytes, -80 dBm	-	85	-	mA
Receive 802.11g, packet length = 1024 bytes, -70 dBm	-	85	-	mA
Receive 802.11n, packet length = 1024 bytes, -65 dBm	-	80	-	mA
Receive 802.11n HT40, packet length = 1024 bytes, -65 dBm	-	80	-	mA

5.4 Wi-Fi Radio

Table 10: Wi-Fi Radio Characteristics

Description	Min	Typical	Max	Unit
Input frequency	2412	-	2484	MHz
Input impedance	-	50	-	Ω
Input reflection	-	-	-10	dB
Output power of PA for 72.2 Mbps	15.5	16.5	17.5	dBm
Output power of PA for 11b mode	19.5	20.5	21.5	dBm
DSSS, 1 Mbps	-	-98	-	dBm
CCK, 11 Mbps	-	-91	-	dBm
OFDM, 6 Mbps	-	-93	-	dBm
OFDM, 54 Mbps	-	-75	-	dBm
HT20, MCS0	-	-93	-	dBm
HT20, MCS7	-	-73	-	dBm
HT40, MCS0	-	-90	-	dBm
HT40, MCS7	-	-70	-	dBm
MCS32	-	-89	-	dBm
OFDM, 6 Mbps	-	37	-	dB
OFDM, 54 Mbps	-	21	-	dB
HT20, MCS0	-	37	-	dB
HT20, MCS7	-	20	-	dB

5.5 Bluetooth Radio

5.5.1 Receiver - Basic Data Rate

Table 11: Receiver Characteristics-Basic Data Rate

Parameter	Conditions	Min	Typ	Max	Unit
Sensitivity @0.1% BER	-	-	-98	-	dBm
Maximum received signal @0.1% BER	-	0	-	-	dBm
Co-channel C/I	-	-	+7	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	-	-6	dB
	F = F0 - 1 MHz	-	-	-6	dB
	F = F0 + 2 MHz	-	-	-25	dB
	F = F0 - 2 MHz	-	-	-33	dB
	F = F0 + 3 MHz	-	-	-25	dB
	F = F0 - 3 MHz	-	-	-45	dB
Out-of-band blocking performance	30 MHz ~ 2000 MHz	-10	-	-	dBm
	2000 MHz ~ 2400 MHz	-27	-	-	dBm
	2500 MHz ~ 3000 MHz	-27	-	-	dBm
	3000 MHz ~ 12.5 GHz	-10	-	-	dBm
Intermodulation	-	-36	-	-	dBm

5.5.2 Transmitter - Basic Data Rate

Table 12: Transmitter Characteristics - Basic Data Rate

Parameter	Conditions	Min	Typ	Max	Unit
RF transmit power	-	-	+4	+4	dBm
RF power control range	-	-	25	-	dB
20 dB bandwidth	-	-	0.9	-	MHz
Adjacent channel transmit power	F = F0 + 1 MHz	-	-24	-	dBm
	F = F0 - 1 MHz	-	-16.1	-	dBm
	F = F0 + 2 MHz	-	-40.8	-	dBm
	F = F0 - 2 MHz	-	-35.6	-	dBm
	F = F0 + 3 MHz	-	-45.7	-	dBm
	F = F0 - 3 MHz	-	-40.2	-	dBm
	F = F0 + > 3 MHz	-	-45.6	-	dBm
	F = F0 - > 3 MHz	-	-44.6	-	dBm
Δ f1$_{avg}$	-	-	-	155	kHz
Δ f2$_{max}$	-	133.7	-	-	kHz
Δ f2$_{avg}$/Δ f1$_{avg}$	-	-	0.92	-	-
ICFT	-	-	-7	-	kHz
Drift rate	-	-	0.7	-	kHz/50 μs
Drift (1 slot packet)	-	-	6	-	kHz
Drift (5 slot packet)	-	-	6	-	kHz

5.5.3 Receiver - Enhanced Data Rate

Table 13: Receiver Characteristics - Enhanced Data Rate

Parameter	Conditions	Min	Typ	Max	Unit
$\pi/4$ DQPSK					
Sensitivity @0.01% BER	-	-	-98	-	dBm
Maximum received signal @0.1% BER	-	-	0	-	dBm
Co-channel C/I	-	-	11	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	-7	-	dB
	F = F0 - 1 MHz	-	-7	-	dB
	F = F0 + 2 MHz	-	-25	-	dB
	F = F0 - 2 MHz	-	-35	-	dB
	F = F0 + 3 MHz	-	-25	-	dB
	F = F0 - 3 MHz	-	-45	-	dB
8DPSK					
Sensitivity @0.01% BER	-	-	-84	-	dBm
Maximum received signal @0.1% BER	-	0	-	-	dBm
C/I c-channel	-	-	18	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	2	-	dB
	F = F0 - 1 MHz	-	2	-	dB
	F = F0 + 2 MHz	-	-25	-	dB
	F = F0 - 2 MHz	-	-25	-	dB
	F = F0 + 3 MHz	-	-25	-	dB
	F = F0 - 3 MHz	-	-38	-	dB

5.5.4 Transmitter - Enhanced Data Rate

Table 14: Transmitter Characteristics - Enhanced Data Rate

Parameter	Conditions	Min	Typ	Max	Unit
Maximum RF transmit power	-	-	+2	-	dBm
Relative transmit control	-	-	-1.5	-	dB
$\pi/4$ DQPSK max w0	-	-	-0.72	-	kHz
$\pi/4$ DQPSK max wi	-	-	-6	-	kHz
$\pi/4$ DQPSK max lwi + w0l	-	-	-7.42	-	kHz
8DPSK max w0	-	-	0.7	-	kHz
8DPSK max wi	-	-	-9.6	-	kHz
8DPSK max lwi + w0l	-	-	-10	-	kHz
$\pi/4$ DQPSK modulation accuracy	RMS DEVM	-	4.28	-	%
	99% DEVM	-	-	30	%
	Peak DEVM	-	13.3	-	%
8 DPSK modulation accuracy	RMS DEVM	-	5.8	-	%
	99% DEVM	-	-	20	%
	Peak DEVM	-	14	-	%

Parameter	Conditions	Min	Typ	Max	Unit
In-band spurious emissions	F = F0 + 1 MHz	-	-34	-	dBm
	F = F0 - 1 MHz	-	-40.2	-	dBm
	F = F0 + 2 MHz	-	-34	-	dBm
	F = F0 - 2 MHz	-	-36	-	dBm
	F = F0 + 3 MHz	-	-38	-	dBm
	F = F0 - 3 MHz	-	-40.3	-	dBm
	F = F0 +/- > 3 MHz	-	-	-41.5	dBm
EDR differential phase coding	-	-	100	-	%

5.6 Bluetooth LE Radio

5.6.1 Receiver

Table 15: Receiver Characteristics - BLE

Parameter	Conditions	Min	Typ	Max	Unit
Sensitivity @0.1% BER	-	-	-98	-	dBm
Maximum received signal @0.1% BER	-	0	-	-	dBm
Co-channel C/I	-	-	+10	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	-5	-	dB
	F = F0 - 1 MHz	-	-5	-	dB
	F = F0 + 2 MHz	-	-25	-	dB
	F = F0 - 2 MHz	-	-35	-	dB
	F = F0 + 3 MHz	-	-25	-	dB
	F = F0 - 3 MHz	-	-45	-	dB
Out-of-band blocking performance	30 MHz ~ 2000 MHz	-10	-	-	dBm
	2000 MHz ~ 2400 MHz	-27	-	-	dBm
	2500 MHz ~ 3000 MHz	-27	-	-	dBm
	3000 MHz ~ 12.5 GHz	-10	-	-	dBm
Intermodulation	-	-36	-	-	dBm

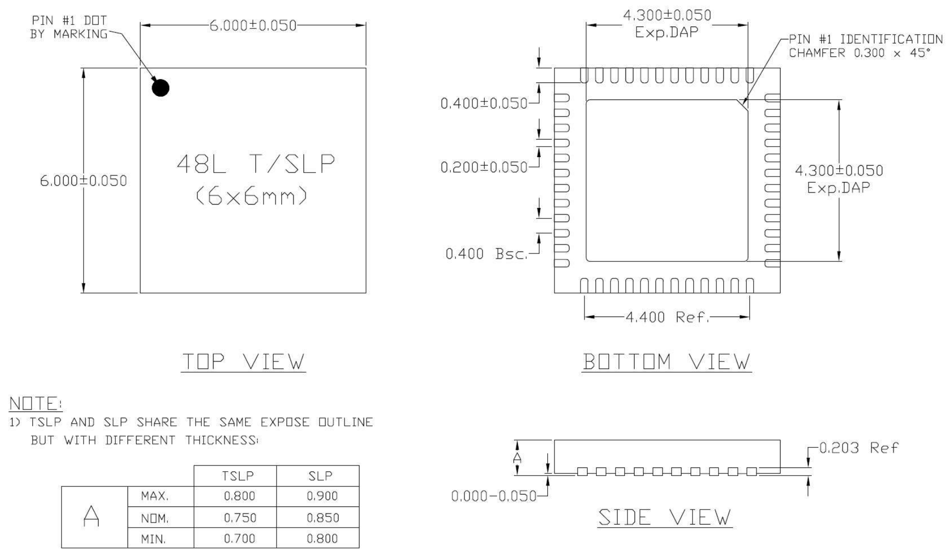
5.6.2 Transmitter

Table 16: Transmitter Characteristics - BLE

Parameter	Conditions	Min	Typ	Max	Unit
RF transmit power	-	-	+7.5	+10	dBm
RF power control range	-	-	25	-	dB
Adjacent channel transmit power	F = F0 + 1 MHz	-	-14.6	-	dBm
	F = F0 - 1 MHz	-	-12.7	-	dBm
	F = F0 + 2 MHz	-	-44.3	-	dBm
	F = F0 - 2 MHz	-	-38.7	-	dBm
	F = F0 + 3 MHz	-	-49.2	-	dBm
	F = F0 - 3 MHz	-	-44.7	-	dBm
	F = F0 + > 3 MHz	-	-50	-	dBm

Parameter	Conditions	Min	Typ	Max	Unit
	F = F0 - > 3 MHz	-	-50	-	dBm
Δ f1$_{avg}$	-	-	-	265	kHz
Δ f2$_{max}$	-	247	-	-	kHz
Δ f2$_{avg}$/Δ f1$_{avg}$	-	-	-0.92	-	-
ICFT	-	-	-10	-	kHz
Drift rate	-	-	0.7	-	kHz/50 μs
Drift	-	-	2	-	kHz

6. Package Information



NOTE:
1) TSLP AND SLP SHARE THE SAME EXPOSE OUTLINE
BUT WITH DIFFERENT THICKNESS:

		TSLP	SLP
A	MAX.	0.800	0.900
	NOM.	0.750	0.850
	MIN.	0.700	0.800

Figure 4: QFN48 (6x6 mm) Package

7. Supported Resources

7.1 Related Documentation

The following link provides related documents of ESP32.

- ESP32 Documentation
  All the available documentation and other resources of ESP32

7.2 Community Resources

The following links connect to ESP32 community resources.

- ESP32 Online Community
  An Engineer-to-Engineer (E2E) Community for ESP32 where you can ask questions, share knowledge, explore ideas and help solve problems with fellow engineers.
- ESP32 Github
  ESP32 development projects are freely distributed under Espressif's MIT license on Github. It is established to help developers get started with ESP32 and foster innovation and the growth of general knowledge about the hardware and software surrounding these devices.

Appendix A - Touch Sensor

A touch sensor system is built on a substrate which carries electrodes and relevant connections with a flat protective surface. When a user touches the surface, the capacitance variation is triggered, and a binary signal is generated to indicate whether the touch is valid.

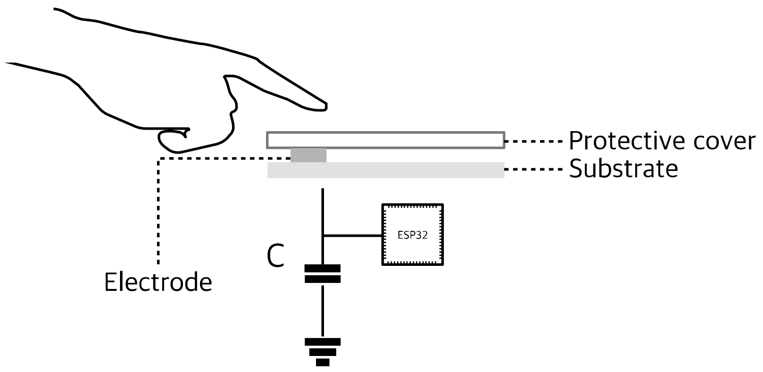


Figure 5: A Typical Touch Sensor Application

In order to prevent capacitive coupling and other electrical interference to the sensitivity of the touch sensor system, the following factors should be taken into account.

A.1. Electrode Pattern

The proper size and shape of an electrode helps improve system sensitivity. Round, oval, or shapes similar to a human fingertip is commonly applied. Large size or irregular shape might lead to incorrect responses from nearby electrodes.

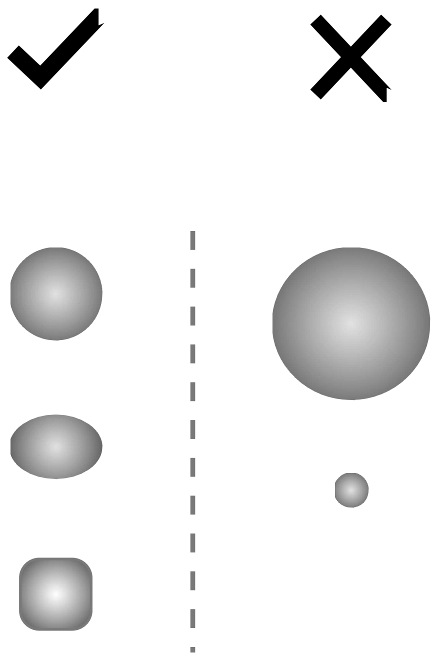


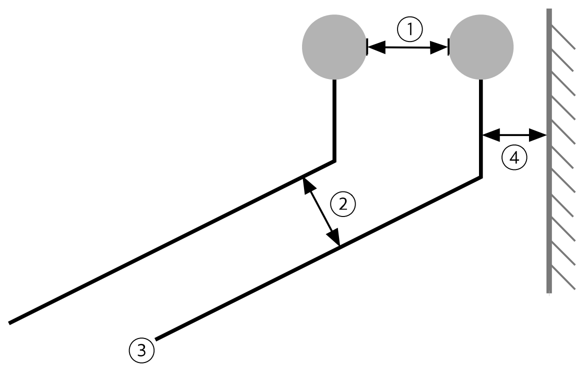
Figure 6: Electrode Pattern Requirements

Note:
The examples illustrated in Figure 6 are not of actual scale. It is suggested that users take a human fingertip as reference.

A.2. PCB Layout

The recommendations for correctly routing sensing tracks of electrodes are as follows:

- Close proximity between electrodes may lead to crosstalk between electrodes and false touch detections. The distance between electrodes should be at least twice the thickness of the panel used.
- The width of a sensor track creates parasitic capacitance, which could vary with manufacturing processes. The thinner the track is, the less capacitive coupling it generates. The track width should be kept as thin as possible and the length should not exceed 10cm to accommodate.
- We should avoid coupling between lines of high frequency signals. The sensing tracks should be routed parallel to each other on the same layer and the distance between the tracks should be at least twice the width of the track.
- When designing a touch sensor device, there should be no components adjacent to or underneath the electrodes.
- Do not ground the touch sensor device. It is preferable that no ground layer be placed under the device, unless there is a need to isolate it. Parasitic capacitance generated between the touch sensor device and the ground degrades sensitivity.



① Distance between electrodes – Twice the thickness of the panel
② Distance between tracks – Twice the track width
③ Width of the track (electrode wiring) – As thin as possible
④ Distance between track and ground plane – 2mm at a minimum

Figure 7: Sensor Track Routing Requirements

Appendix B - Code Examples

B.1. Input

```
>python esptool.py -p dev/tty8 -b 115200 write_Flash -c ESP32 -ff 40m -fm qio -fs 2MB
0x0 ~/Workspace/ESP32_BIN/boot.bin
0x04000 ~/Workspace/ESP32_BIN/drom0.bin
0x40000 ~/Workspace/ESP32_BIN/bin/irom0_Flash.bin
0xFC000 ~/Workspace/ESP32_BIN/blank.bin
0x1FC000 ~/Workspace/ESP32_BIN/esp_init_data_default.bin
```

B.2. Output

```
Connecting...
Erasing Flash...
Wrote 3072 bytes at 0x00000000 in 0.3 seconds (73.8 kbit/s)...
Erasing Flash...
Wrote 395264 bytes at 0x04000000 in 43.2 seconds (73.2 kbit/s)...
Erasing Flash...
Wrote 1024 bytes at 0x40000000 in 0.1 seconds (74.5 kbit/s)...
Erasing Flash...
Wrote 4096 bytes at 0xfc000000 in 0.4 seconds (73.5 kbit/s)...
Erasing Flash...
Wrote 4096 bytes at 0x1fc00000 in 0.5 seconds (73.8 kbit/s)...
Leaving...
```