**VIETNAM GENERAL CONFEDERATION OF LABOUR**

**TON DUC THANG UNIVERSITY**

**FACULTY OF ELECTRICAL & ELECTRONICS ENGINEERING**

****

**NGUYEN DUC TRUNG DAN**

**CONTROL CAR**

**INDIVIDUAL PROJECTS**

**AUTOMATION AND CONTROL ENGINEERING**

**HO CHI MINH CITY, YEAR 2022**

**VIETNAM GENERAL CONFEDERATION OF LABOUR**

**TON DUC THANG UNIVERSITY**

**FACULTY OF ELECTRICAL & ELECTRONICS ENGINEERING**

****

**NGUYEN DUC TRUNG DAN**

**CONTROL CAR**

**INDIVIDUAL PROJECTS**

**AUTOMATION AND CONTROL ENGINEERING**

***Under the supervision of***

**Dr. Vo Hoang Duy**

**HO CHI MINH CITY, YEAR 2022**

**ACKNOWLEDGMENT**

During my study at Ton Duc Thang University, I was enthusiastically taught and imparted useful knowledge by teachers so that I could gain very important knowledge for my major. mine later. In fact, there is no success that is not tied to the support and help directly or indirectly from others. I would like to thank the school administration and the teachers who dedicatedly taught me to complete the subject well. I would also like to thank my parents, who raised, cared for, and gave me good education.

I wish the teachers more and more health to strive for high achievements in teaching work. Wishing you and your family good health, good health, happiness and lots of joy. Wish Ton Duc Thang University will always be a belief and a solid support for many generations of students with their learning journey.

Thank you sincerely!

*Ho Chi Minh City, day 30 month 4 year 2023*

*Author*

*Dan*

*Nguyen Duc Trung Dan*

**DISCLAIMER OF SPECIALIZED PROJECTS**

The work was completed at Ton Duc Thang University

Course instructor Dr. Vo Hoang Duy

*(Title, full name and signature)*

This thesis is defended at the Undergraduate Thesis Examination Committee was hold at Ton Duc Thang University on … /…/……

Confirmation of the Chairman of the Undergraduate Thesis Examination Committee and the Dean of the faculty after receiving the modified thesis (if any)

**CHAIRMAN DEAN OF FACULTY**

**…………………………. ………………………………**

**DECLARATION OF AUTHORSHIP**

I hereby declare that this thesis was carried out by myself under the guidance and supervision of Dr. Vo Hoang Duy; and that the work and the results contained in it are original and have not been submitted anywhere for any previous purposes. The data and figures presented in this thesis are for analysis, comments, and evaluations from various resources by my own work and have been duly acknowledged in the reference part.

In addition, other comments, reviews, and data used by other authors, and organizations have been acknowledged, and explicitly cited.

I will take full responsibility for any fraud detected in my thesis. Ton Duc Thang University is unrelated to any copyright infringement caused on my work (if any).

*Ho Chi Minh City, day 30 month 4 year 2023*

*Author*

*Dan*

*Nguyen Duc Trung Dan*

**CONTENTS**

[**LIST OF FIGURES** 8](#_Toc134308631)

[**LIST OF TABLES** 9](#_Toc134308632)

[**CHAPTER 1. OVERVIEW** 1](#_Toc134308633)

[**1.1** **Introduction to the topic** 1](#_Toc134308634)

[**1.2** **Research purposes** 1](#_Toc134308635)

[**1.3** **Research object** 1](#_Toc134308636)

[**1.4** **Research scope** 1](#_Toc134308637)

[**1.5** **Expect results.** 1](#_Toc134308638)

[**CHAPTER 2. THEORETICAL BASIS** 3](#_Toc134308639)

[**2.1 Introduction of ESP32-DevKitC use module ESP-WROOM-32** 3](#_Toc134308640)

[**2.2 Introduction of L298N DC Motor Driver Module** 13](#_Toc134308641)

[**2.3 Introduction of servo motor SG90** 17](#_Toc134308642)

[**2.4 Introduction of gear motor yellow.** 20](#_Toc134308643)

[**2.5 Introduction of Ultrasonic sensor module SRF05** 21](#_Toc134308644)

[**2.6 Introduction to PMW Pulse** 23](#_Toc134308645)

[**2.7 Introduction to MIT App Inventor 2** 25](#_Toc134308646)

[**CHAPTER 3. ALGORITHMS AND CONTROLS** 28](#_Toc134308647)

[**3.1 System Operation** 28](#_Toc134308648)

[**3.2 Algorithm flowchart on microcontroller** 28](#_Toc134308649)

[**CHAPTER 4. DESIGN AND CONSTRUCTION** 29](#_Toc134308650)

[**4.1 Block diagram of the system** 29](#_Toc134308651)

[**4.1.1 Control block** 29](#_Toc134308652)

[**4.1.2 Processing Block** 29](#_Toc134308653)

[**4.1.3 Sensor Block** 30](#_Toc134308654)

[**4.2 Model Design** 31](#_Toc134308655)

[**4.2.1 Design APP on MIT inventor** 31](#_Toc134308656)

[**4.2.2. Logical connection diagram** 32](#_Toc134308657)

[**4.2.3 App in mobie phone** 33](#_Toc134308658)

[**4.3 Experiment** 34](#_Toc134308659)

[**4.3.1 Implementation process** 34](#_Toc134308660)

[**4.3.2 Results** 34](#_Toc134308661)

[**4.3.3 Experiment overview** 34](#_Toc134308662)

[**CHAPTER 5. CONCLUSION** 35](#_Toc134308663)

[**5.1 Advantage** 35](#_Toc134308664)

[**5.2 Disadvantage** 35](#_Toc134308665)

[**5.3 Development** 35](#_Toc134308666)

[**REFERENCES** 36](#_Toc134308667)

[**APPENDIX 1** 37](#_Toc134308668)

# **LIST OF FIGURES**

[Figure 1:PIN OUT ESP32 5](#_Toc134308705)

[Figure 2: GPIO PINS 6](#_Toc134308706)

[Figure 3: ADC PINS 7](#_Toc134308707)

[Figure 4: DAC PINS 8](#_Toc134308708)

[Figure 5: TOUCH PINS 8](#_Toc134308709)

[Figure 6: I2C PINS 9](#_Toc134308710)

[Figure 7: SPI PINS 9](#_Toc134308711)

[Figure 8: UART PIN 10](#_Toc134308712)

[Figure 9: PWM PINS 11](#_Toc134308713)

[Figure 10: RTC GPIO PINS 11](#_Toc134308714)

[Figure 11:STRAPPING PINS 12](#_Toc134308715)

[Figure 12: POWER PIN 12](#_Toc134308716)

[Figure 13: ENABLE PIN 13](#_Toc134308717)

[Figure 14: Light Sensor BH1750 14](#_Toc134308718)

[Figure 15: L298N PINS 15](#_Toc134308719)

[Figure 16: PMW SIGNAL 16](#_Toc134308720)

[Figure 17: OPERATION PRINCIPLE OF THE BRIDGE CIRCUIT 17](#_Toc134308721)

[Figure 18: SERVO 18](#_Toc134308722)

[Figure 19: PRINCIPLE OF SERVO 19](#_Toc134308723)

[Figure 20: SG90 Servo Motor Dimensions 20](#_Toc134308724)

[Figure 21: GEAR MOTOR 20](#_Toc134308725)

[Figure 22: Ultrasonic ultrasonic sensor 21](#_Toc134308726)

[Figure 23: ACTIVE MODE 1 OF SENSOR 23](#_Toc134308727)

[Figure 24:: ACTIVE MODE 2 OF SENSOR 23](#_Toc134308728)

[Figure 25: INTERVIEW AT MIT APP 26](#_Toc134308729)

[Figure 26: INTERFAVE APP MIT 27](#_Toc134308730)

[Figure 27: FUNCTION LOGIC OF MIT 27](#_Toc134308731)

[Figure 28: Block 29](#_Toc134308732)

[Figure 29: WORKING DISPLAY 1 31](#_Toc134308733)

[Figure 30:WORKING DISPLAY 2 31](#_Toc134308734)

[Figure 31:DIAGRAM OF VOICE CONTROL 32](#_Toc134308735)

[Figure 32:DIAGRAM OF REMOTE CONTROL 32](#_Toc134308736)

[Figure 33: Screen 1 in phone 33](#_Toc134308737)

[Figure 34: Screen 2 in phone 33](#_Toc134308738)

# **LIST OF TABLES**

[Table 1: SERVO PINS 19](#_Toc134308739)

# **CHAPTER 1. OVERVIEW**

## **Introduction to the topic**

* Topic name: “Voice Control Car”
* Request object: Program the ESP 32 to control 4 servo motor SG90 and 4 gear motor through module L298 DC Motor Driver. Display Control ​​on smartphone.
* Project implementation direction:

**-** Using ESP32-WROOM-32 is the control center and using Bluetooth function. Use the Arduino IDE program to program C and compile the program to control.

- Use 4 servo motor SG90 and 4 gear motor through module L298 DC Motor Driver

- Use MIT app developer to create a mobile app that controls the middleware via Bluetooth.

- Use ultrasonic sensor module SRF05

- Three mode: Voice control, Manual control and auto control.

## **Research purposes**

* Apply what you have learned in programming.
* Understand how the L298 DC Motor Driver and servo motor SG90 works.
* Understand how to calculate ultrasonic sensor module SRF05 and work this.
* Understand how to program and use software algorithms MIT.
* Understand how to use PMW signals.
* Esp32 chip research

## **Research object**

* ESP32-DevKitC
* L298 DC Motor Driver
* Servo motor SG90
* Gear motor
* Ultrasonic sensor module SRF05
* Write associative logic function in MIT

## **Research scope**

Dynamic voice control via Bluetooth, create a control panel, know how to use the MIT inventor app.

## **Expect results.**

Stable Bluetooth connection, execute commands as desired, motors run well as required and the car automatically fires and dodges obstacles.

# **CHAPTER 2. THEORETICAL BASIS**

## **2.1 Introduction of ESP32-DevKitC use module ESP-WROOM-32**

* ESP32-DevKitC is an entry-level development board for the ESP32 microcontroller chip 1, produced by Espressif. It has many of the ESP32 pins exposed, making it easy to connect to other components, and it features integrated flash memory, USB-to-serial converter, and started resistors for easier development, testing and prototyping. It is designed to support a wide range of peripheral devices and sensors, making it a popular choice for prototyping and development of Internet of Things (IoT) applications. It has several variants and is widely available from different vendors.
* Moudule ESP32-WROOM-32 is a single-board microcontroller and module, based on the ESP32 microcontroller chip, that provides Wi-Fi and Bluetooth connectivity. It includes integrated flash memory, making it a suitable choice for a wide range of applications, from low-power sensor networks to robust Internet of Things (IoT) applications. It is designed to be used in a wide range of applications and is a low-power and low-cost solution for Internet of Things (IoT) projects. Several development boards based on ESP32-WROOM-32 are available from different vendors, which provide various options for creating IoT projects.
* It is a successor to the ESP8266 microcontroller.
* Language environment we can use:
* Arduino IDE
* Espressif IDF (IoT Development Framework)
* MicroPython
* JavaScript
* LUA
* v.v.

**ESP32-DevKitC use module ESP-WROOM-32*: PHYSICAL COMPONENTS***

#*ESP-WROOM-32 microcontroller*

* ***SPECIFICATIONS***
* Size: 18 mm x 20 mm x 3 mm
* CPU: XTENSA Dual-Core 32-bit LX6 with operating frequency up to 240 MHz
* Internal memory:

+ 448 KBYTES ROM for booting and features of the chip core.

+ 520 KBYTES SRAM on the chip for data and instruction commands.

+ 8 Kbytes SRAM in RTC (called RTC Slow Memory) for access by Co-Processor sets

+ 8 Kbytes SRAM in RTC (called RTC Fast Memory) for data, access by CPU when RTC is boot from Deep-Sleep mode.

+ 1 Kbit Efuse, with 256 bits for the system (MAC address and chip configuration), the remaining 768 is for the user application, including Flash memory encryption and ID for chips.

* Wifi connection:

+ Wi-Fi: 802.11 b/g/n/e/i

+ Bluetooth: Br/Edr V4.2 and Ble versions

* Ethernet Mac supports standard: DMA and IEEE 1588
* Bus supports carrying can 2.0
* Peripheral communication:

+ 12 -bit ADC converter, 16 channels

+ 8-bit DAC converter: 2 channels

+ 10 legs to communicate with touch sensor (touch sensor)

+ IR (TX/RX)

+ PWM output for motor control

+ LED PWM: 16 channels

+ Hall sensor

+ Temperature sensor

+ 4 x spi

+ x I²S

+ x i²C

+ x UART

* Stable operation temperature: -40C to 85C
* Operating voltage: 2.2-3.6V
* Stable consumption line: 80mA
* **SECURITY**
* IEEE 802.11 supports security standards: WFA, WPA/WPA2 and WAPI
* Flash encryption
* 1024-bit otp, 768-bit for users
* **PIN MODE**

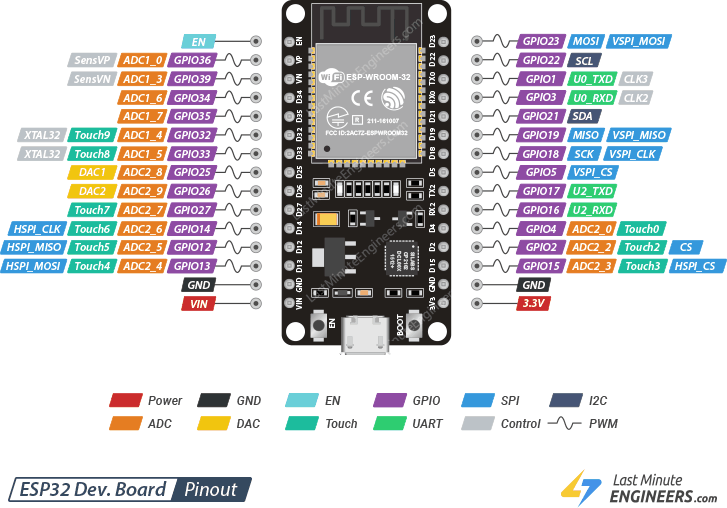


Figure :PIN OUT ESP32

* **ESP32 Peripherals and I/O**

*Including:*

* Graphical user interface, text, application, email

  Description automatically generated**GPIO PINS**

The ESP32 development board has 25 GPIO pins that can be assigned different functions by programming the appropriate registers. There are several kinds of GPIOs: digital-only, analog-enabled, capacitive-touch-enabled, etc. Analog-enabled GPIOs and Capacitive-touch-enabled GPIOs can be configured as digital GPIOs. Most of these digital GPIOs can be configured with internal pull-up or pull-down, or set to high impedance.

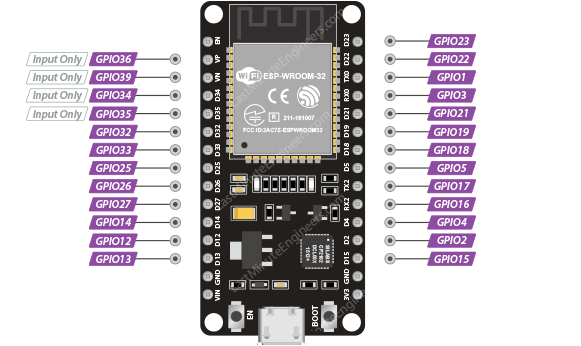


Figure : GPIO PINS

* **ADC PINS**
* ESP32 integrates two 12-bit SAR ADCs and supports measurements on 15 channels (analog-enabled pins).

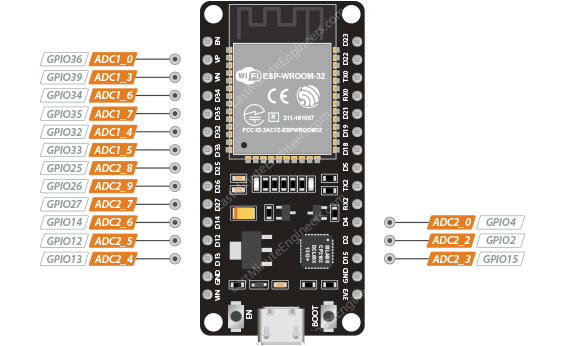


Figure : ADC PINS

* The ESP32’s ADC is a 12-bit ADC, which means it can detect 4096 (2^12) discrete analog levels. In other words, it will convert input voltages ranging from 0 to 3.3V (operating voltage) into integer values ranging from 0 to 4095. This results in a resolution of 3.3 volts / 4096 units, or 0.0008 volts (0.8 mV) per unit.
* **DAC PINS**
* The ESP32 includes two 8-bit DAC channels for converting digital signals to true analog voltages. It can be used as a “digital potentiometer” to control analog devices.
* These DACs have an 8-bit resolution, which means that values ranging from 0 to 256 will be converted to an analog voltage ranging from 0 to 3.3V.

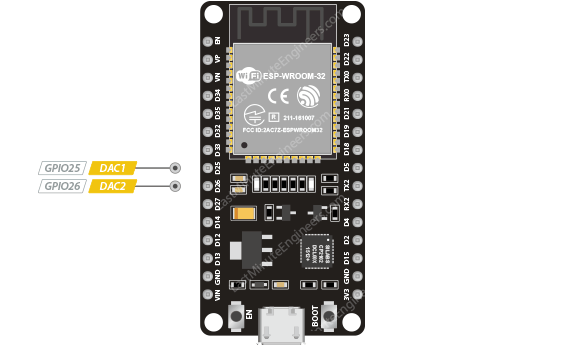


Figure : DAC PINS

* **TOUCH PINS**

The ESP32 has 9 capacitive touch-sensing GPIOs. When a capacitive load (such as a human finger) is in close proximity to the GPIO, the ESP32 detects the change in capacitance.

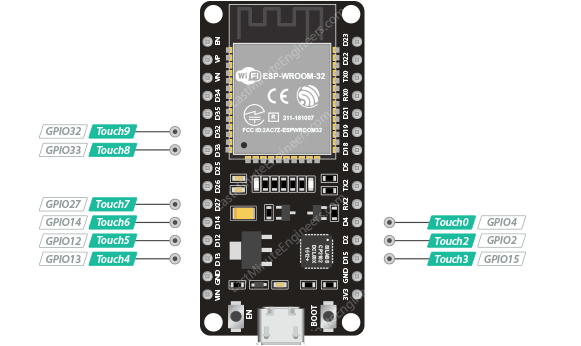


Figure : TOUCH PINS

* **I2C PINS**
* The ESP32 has a single I2C bus that allows you to connect up to 112 sensors and peripherals. The SDA and SCL pins are, by default, assigned to the following pins. However, you can bit-bang the I2C protocol on any GPIO pins with the wire.begin(SDA, SCL) command.

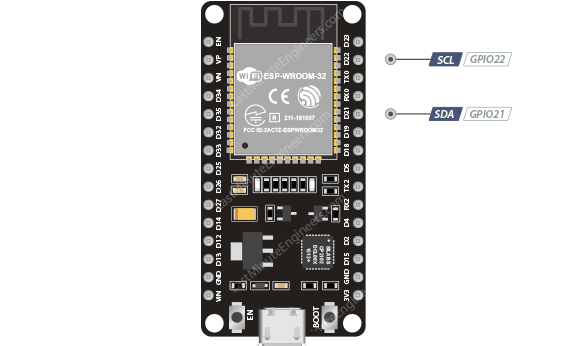


Figure : I2C PINS

* **SPI PINS**

- ESP32 features three SPIs (SPI, HSPI, and VSPI) in slave and master modes. These SPIs also support the general-purpose SPI features listed below:

* 4 timing modes of the SPI format transfer
* Up to 80 MHz and the divided clocks of 80 MHz
* Up to 64-Byte FIFO

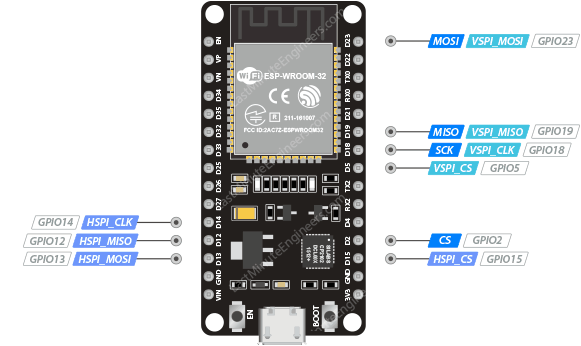


Figure : SPI PINS

* **UART PIN**
* The ESP32 dev. board has three UART interfaces, UART0, UART1, and UART2, that support asynchronous communication (RS232 and RS485) and IrDA at up to 5 Mbps.
* UART0 pins are connected to the USB-to-Serial converter and are used for flashing and debugging. Therefore, the UART0 pins are not recommended for use.
* UART1 pins are reserved for the integrated flash memory chip.
* UART2, on the other hand, is a safe option for connecting to UART-devices such as GPS, fingerprint sensor, distance sensor, and so on.

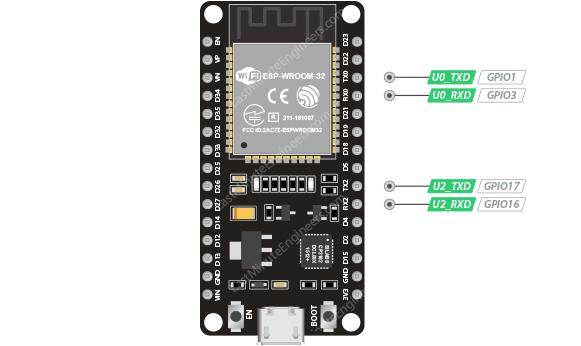


Figure : UART PIN

* **PMW PINS**
* The board has 21 channels (all GPIOs except input-only GPIOs) of PWM pins controlled by a PWM controller. The PWM output can be used for driving digital motors and LEDs. The board has 21 channels (all GPIOs except input-only GPIOs) of PWM pins controlled by a PWM controller. The PWM output can be used for driving digital motors and LEDs.

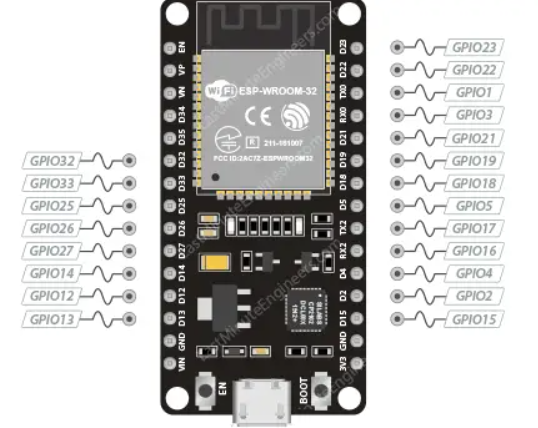


Figure : PWM PINS

* **RTC GPIO PINS**
* Some GPIOs are routed to the RTC low-power subsystem and are referred to as RTC GPIOs. These pins are used to wake the ESP32 from deep sleep when the Ultra Low Power (ULP) co-processor is running. The GPIOs highlighted below can be used as external wake up sources.

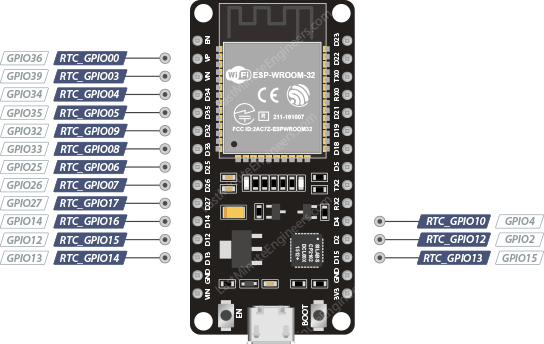


Figure : RTC GPIO PINS

* **STRAPPING PINS**
* These pins are used to put the ESP32 into BOOT mode (to run the program stored in the flash memory) or FLASH mode (to upload the program to the flash memory). Depending on the state of these pins, the ESP32 will enter BOOT mode or FLASH mode at power on.

A picture containing text, electronics

Description automatically generated

Figure :STRAPPING PINS

* **POWER PINS**
* There are two power pins: the VIN pin and the 3V3 pin. The VIN pin can be used to directly power the ESP32 and its peripherals, if you have a regulated 5V power supply. The 3V3 pin is the output from the on-board voltage regulator; you can get up to 600mA from it. GND is the ground pin.

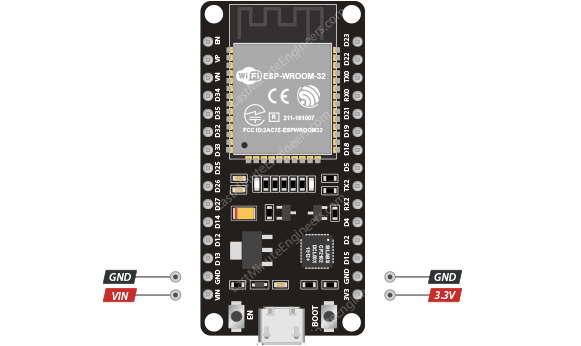


Figure : POWER PIN

* **ENABLE PIN**
* The EN pin is the enable pin for the ESP32, pulled high by default. When pulled HIGH, the chip is enabled; when pulled LOW, the chip is disabled.
* The EN pin is also connected to a pushbutton switch that can pull the pin LOW and trigger a reset.

A picture containing text, electronics, circuit

Description automatically generated

Figure : ENABLE PIN

## **2.2 Introduction of L298N DC Motor Driver Module**

L298N module is a high voltage, high current dual full-bridge motor driver module for controlling DC motor and stepper motor. It can control both the speed and rotation direction of two DC motors. This module consists of an L298 dual-channel H-Bridge motor driver IC. This module uses two techniques for the control speed and rotation direction of the DC motors. These are PWM – For controlling the speed and H-Bridge

For controlling rotation direction. These modules can control two DC motor or one stepper motor at the same time.

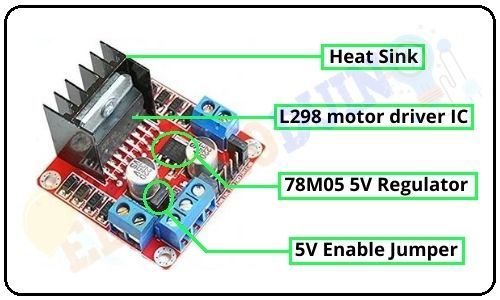


Figure 14: Light Sensor BH1750

***L298N Motor Driver Features:***

* The main features are discussed here
* Its input voltage value is 3.2 volts to 40VDc
* It is L298N Dual H Bridge DC Motor Driver board
* Its hugest current is two Amperes
* DC volts from the power supply are five to thirty-five volts
* The operating current is zero to thirty-six milliamperes
* It has power on LED indicator on board
* It comes with a heatsink to minimize the heat losses
* The logic voltage for this device is five volts
* The highest power is twenty-five watts
* Storage temperature is -25 ℃ to +130 ℃.
* Board dimensions are 3.4cm x 4.3cm x 2.7cm

***L298N Module Pinout :***

* The main pinout of L298N is explained here
* IN1, IN2: these pins are input pins of motor A control the direction of this motor
* In3, IN4: Used for motor B direction control and its inputs
* ENA: Enable pin pulse width modulation for Motor A
* ENB: enable pin for motor B
* Out1, Out2: used for output of motor A
* Out3. out4: o/p pins of motor B
* 12V: It is 12 volts input pin that connected to DC supply.
* 5V: It power to switch circuits
* GND: Ground pinout

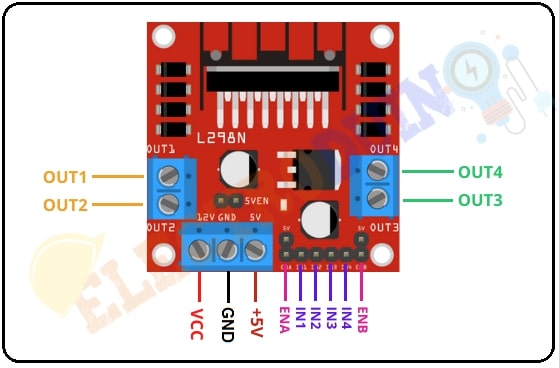


Figure : L298N PINS

* ***How Motor Driver Module Works***

We can only have full control over a DC motor if we can control its speed and spinning direction. This is possible by combining these two techniques.

* PWM – to control speed.
* H-Bridge – to control the spinning direction.

**PWM – to control speed:**

* The speed of a DC motor can be controlled by changing its input voltage. A widely used technique to accomplish this is Pulse Width Modulation (PWM).
* PWM is a technique in which the average value of the input voltage is adjusted by sending a series of ON-OFF pulses. This average voltage is proportional to the width of the pulses, which is referred to as the Duty Cycle.
* The higher the duty cycle, the higher the average voltage applied to the DC motor, resulting in an increase in motor speed. The shorter the duty cycle, the lower the average voltage applied to the DC motor, resulting in a decrease in motor speed.
* The image below shows PWM technique with various duty cycles and average voltages.

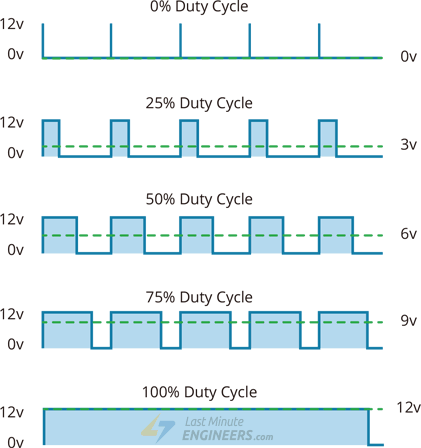


Figure : PMW SIGNAL

**H-Bridge – to control the spinning direction.**

* The spinning direction of a DC motor can be controlled by changing the polarity of its input voltage. A widely used technique to accomplish this is to use an H-bridge.
* An H-bridge circuit is made up of four switches arranged in a H shape, with the motor in the center.
* Closing two specific switches at the same time reverses the polarity of the voltage applied to the motor. This causes a change in the spinning direction of the motor.
* The following animation shows the working of the H-bridge circuit.

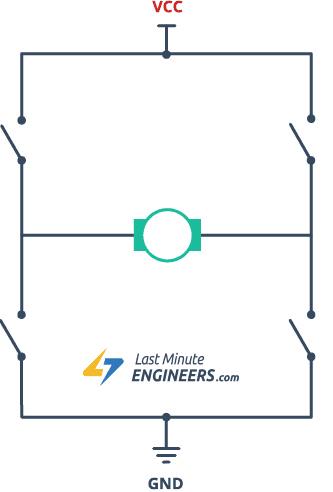


Figure : OPERATION PRINCIPLE OF THE BRIDGE CIRCUIT

## **2.3 Introduction of servo motor SG90**

A servo motor is a type of motor that can rotate with great precision. Normally this type of motor consists of a control circuit that provides feedback on the current position of the motor shaft, this feedback allows the servo motors to rotate with great precision.



Figure : SERVO

***TowerPro SG-90 Features***

* Operating Voltage is +5V typically
* Torque: 2.5kg/cm
* Operating speed is 0.1s/60°
* Gear Type: Plastic
* Rotation : 0°-180°
* Weight of motor : 9gm
* Package includes gear horns and screws



**Servo Motor Wire Configuration**

|  |  |  |
| --- | --- | --- |
| **Wire Number** | **Wire Color** | **Description** |
| 1 | Brown | Ground wire connected to the ground of system |
| 2 | Red | Powers the motor typically +5V is used |
| 3 | Orange | PWM signal is given in through this wire to drive the motor |

Table : SERVO PINS

* **How to use a Servo Motor**

After selecting the right Servo motor for the project, comes the question how to use it. As we know there are three wires coming out of this motor. The description of the same is given on top of this page. To make this motor rotate, we have to power the motor with +5V using the Red and Brown wire and send PWM signals to the Orange color wire. Hence we need something that could generate PWM signals to make this motor work, this something could be anything like a 555 Timer or other Microcontroller platforms like Arduino, PIC, ARM or even a microprocessor like Raspberry Pie. Now, how to control the direction of the motor? To understand that let us a look at the picture given in the datasheet.

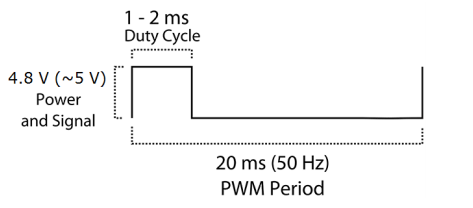


Figure : PRINCIPLE OF SERVO

From the picture we can understand that the PWM signal produced should have a frequency of 50Hz that is the PWM period should be 20ms. Out of which the On-Time can vary from 1ms to 2ms. So when the on-time is 1ms the motor will be in 0° and when 1.5ms the motor will be 90°, similarly when it is 2ms it will be 180°. So, by varying the on-time from 1ms to 2ms the motor can be controlled from 0° to 180°

* **Applications**
* Used as actuators in many robots like Biped Robot, Hexapod, robotic arm etc..
* Commonly used for steering system in RC toys
* Robots where position control is required without feedback
* Less weight hence used in multi DOF robots like humanoid robots

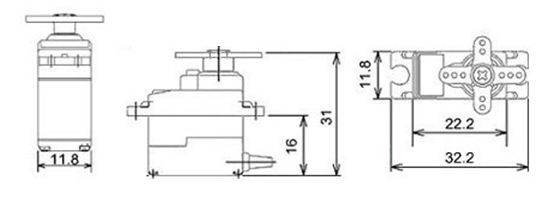


Figure : SG90 Servo Motor Dimensions

## **2.4 Introduction of gear motor yellow.**

This yellow DC motor is ideal for robotics and model vehicles. It has a 1:48 gear ratio and maximum torque of 800g/cm at a minimum of 3V.



Figure : GEAR MOTOR

**FEATURES**

* Perfect for DIY hobby robotics & model vehicles
* 1:48 gear ratio
* Max. 800g/cm of torque at a minimum of 3V

**SPECIFICATIONS**

* Operating Voltage3V-12VDC
* Maximum Torque800g/cm max. @ 3V
* Gear Ratio1:48
* Load Current70mA (250mA max. @ 3V)
* Weight29g
* Dimensions
* Length65 mm
* Width37 mm
* Height22 mm

## **2.5 Introduction of Ultrasonic sensor module SRF05**

Ultrasonic ultrasonic sensor HY-SRF05 is often used in connection with the arduino circuit: emitting ultrasonic waves and receiving ultrasonic waves to reflect back when encountering obstacles. Used in measuring distances, or used to detect and avoid obstacles.

Used to measure distances, measure liquid levels, track robots, line detectors to avoid obstacles...

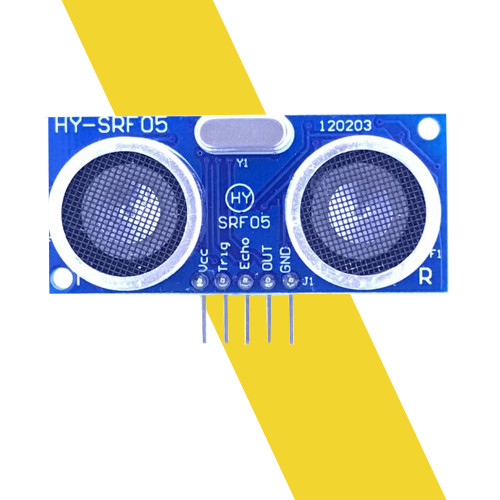


Figure : Ultrasonic ultrasonic sensor

**TECHNICAL INFORMATION**

* Operating voltage: 5VDC
* Play distance: 2cm – 450cm
* Accuracy: ±0.2cm
* Input signal: 10us conflict TTL
* Pinout: there are 5 pins
* VCC : 5Vdc
* Trig(T): digital input.
* echo (R): digital output.
* OUTSIDE
* GND

Ultrasonic sensor HY-SRF05 has two uses: using Echo / Trigger pin pairs or using only 1 Out pin to transmit and receive signals, the sensor is commonly used with countless libraries and Code sets. sample with Arduino.

* **WORKING PRINCIPLE**

**+ In mode 1**: Isolation, trigger and response To measure the distance, we emit a very short pulse (5 Microseconds) from the TRIG pin. The sensor will then generate a HIGH pulse at the ECHO pin until it receives a reflected pulse at this pin. The width of the pulse will be equal to the time the ultrasonic wave is emitted from the sensor back. The speed of sound in air is 340 m/s which is equivalent to 29,412 Microseconds/cm(1000000/(340\*100)). Once we have calculated the time we divide by 29,412 to get the distance to be measured.

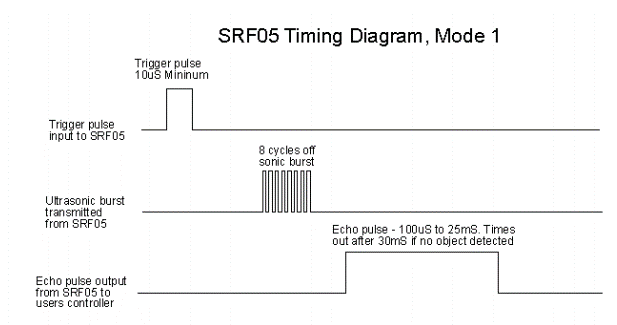


Figure : ACTIVE MODE 1 OF SENSOR

**+ In mode 2**: Use 1 pin for both trigger and response.

We use the OUT pin so that it just emits a pulse and then receives a reflected pulse, the mode pin is grounded. The feedback signal will appear on the same pin as the trigger signal. SR05 will not increase the feedback current for to 700uS after the end of the trigger signals and you've had time to trigger the pin around and make it 1 input.

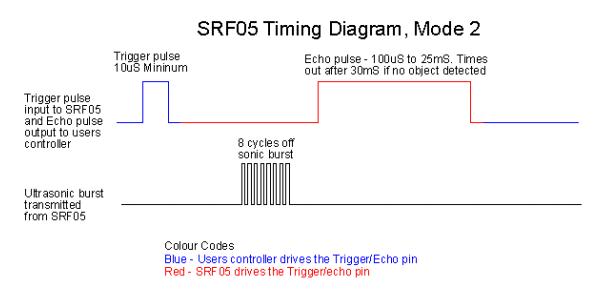
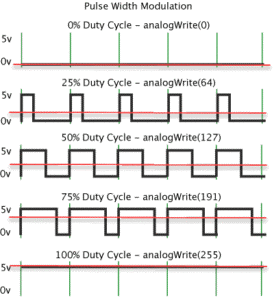


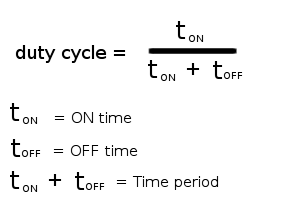
Figure :: ACTIVE MODE 2 OF SENSOR

## **2.6 Introduction to PMW Pulse**

PWM (Pulse Width Modulation) is to modulate (change) the width of the pulse (the time difference between the "ON" and "OFF" pulses) without changing the frequency of the electrical signal.

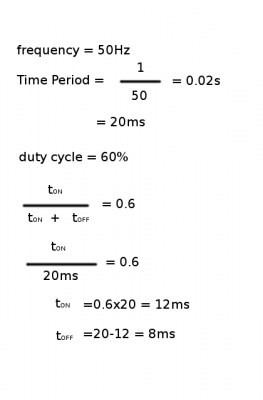
****

* *Pulse modulation calculation formula*

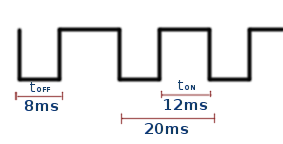
****

* *How to generate PWM modulation pulse*

For example at a frequency of 50Hz design a PWM modulated signal with a duty cycle of 60%.

****

The PWM signal now has the following form:

****

So to generate such a pulse sequence, the programmer only needs to write a program for the microcontroller. Every 8ms the output pulls up 1, then at 1 exactly 12ms pulls the output down to 0. Just like that, with a cycle of 20ms repeating that pulse, we can create the desired pulse sequence with frequency 50Hz and equal frequency. modulation depth is 60%

## **2.7 Introduction to MIT App Inventor 2**

App Inventor lets you develop applications for Android phones using a web browser and either a connected phone or emulator. The App Inventor servers store your work and help you keep track of your projects.

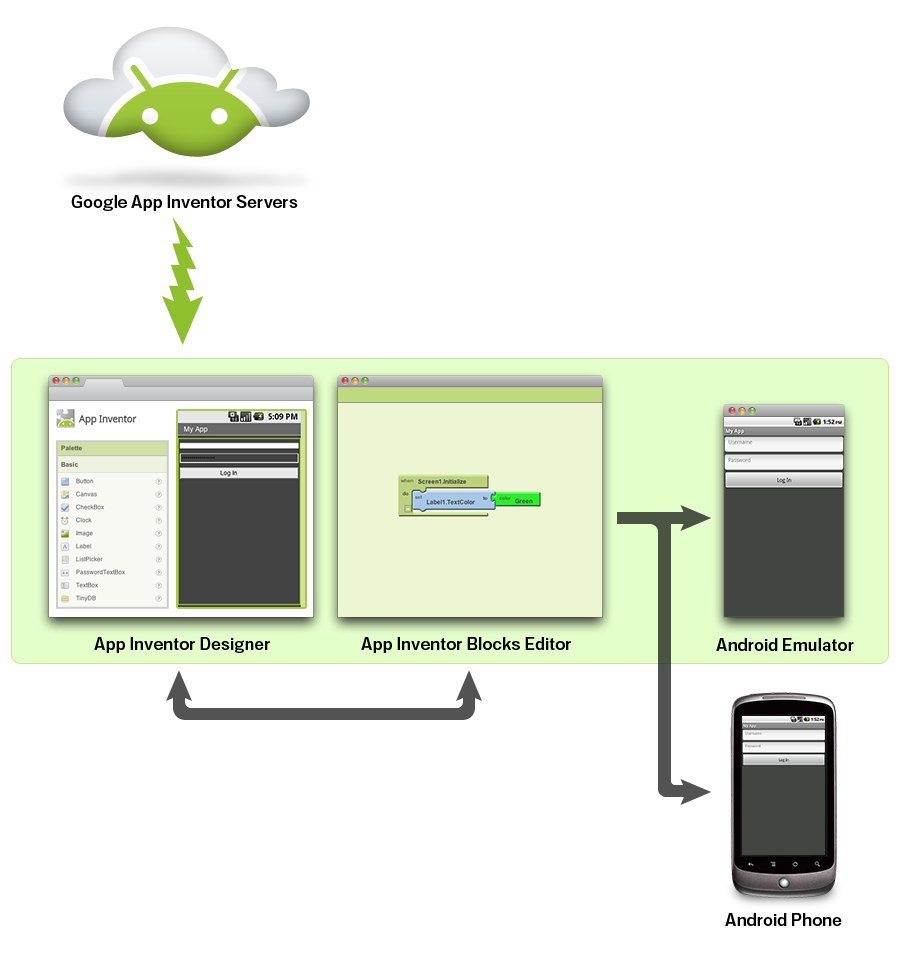


Figure : INTERVIEW AT MIT APP

MIT App Inventor for Android is an open source web application originally provided by Google and currently maintained by the Massachusetts Institute of Technology (MIT). The platform allows developers to create software applications for the Android operating system (OS). Using a graphical interface, the platform allows users to drag and drop blocks of code to create applications that can run on Android devices. As of now, 07/2017, the iOS version of this platform has begun to be tested by Thinkable, which is one of the web application providers for this language.

The core goal of MIT App Inventor is to help people who have no prior programming language knowledge to create useful applications on the Android operating system. The latest version is MIT App Inventor 2

* **THE INTERFACE OF THE APP**

**A screen shot of a cell phone

Description automatically generated with medium confidence**

Figure : INTERFAVE APP MIT

* ***Connect according to the logic of drag and drop and option selection***

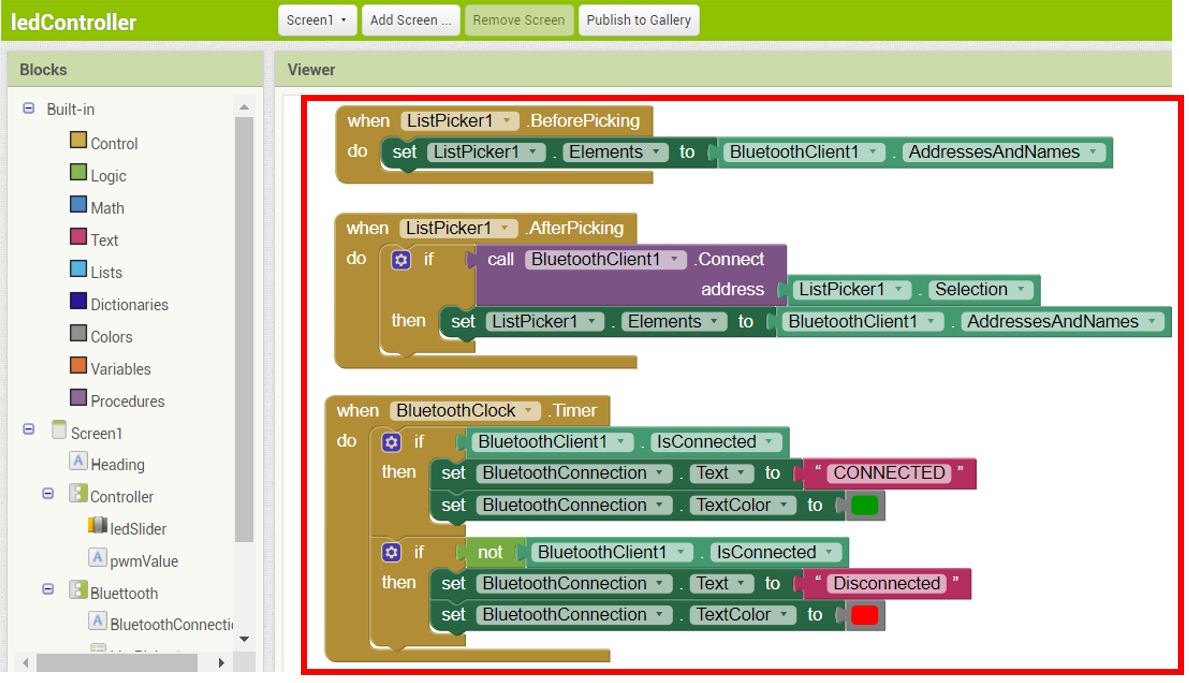


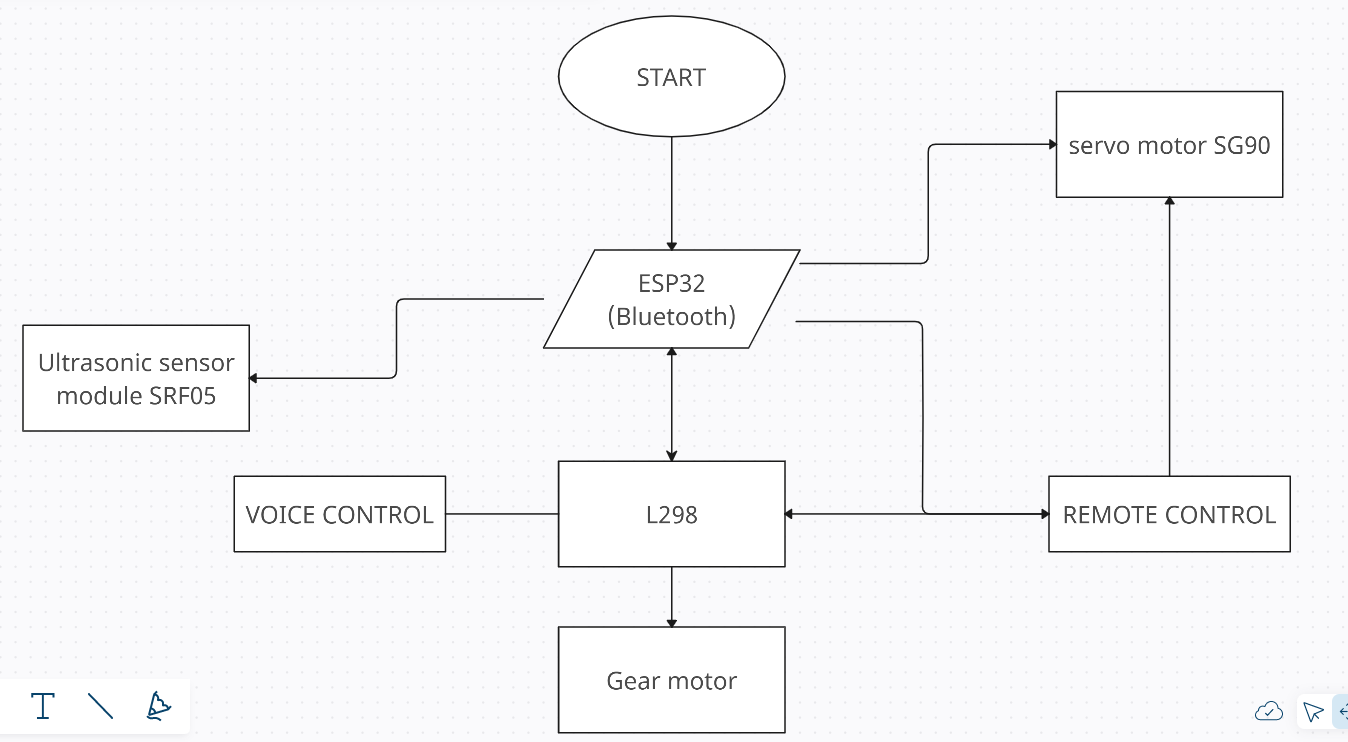
Figure : FUNCTION LOGIC OF MIT

# **CHAPTER 3. ALGORITHMS AND CONTROLS**

## **3.1 System Operation**

We issue control commands via Bluetooth using a mobile app created with the MIT app. When there is a voice signal, it will transmit the signal through the MIT app and switch to the ESP32 Devkit microcontroller to control the motor as desired.

## **3.2 Algorithm flowchart on microcontroller**



# **CHAPTER 4. DESIGN AND CONSTRUCTION**

## **4.1 Block diagram of the system**

Diagram

Description automatically generated

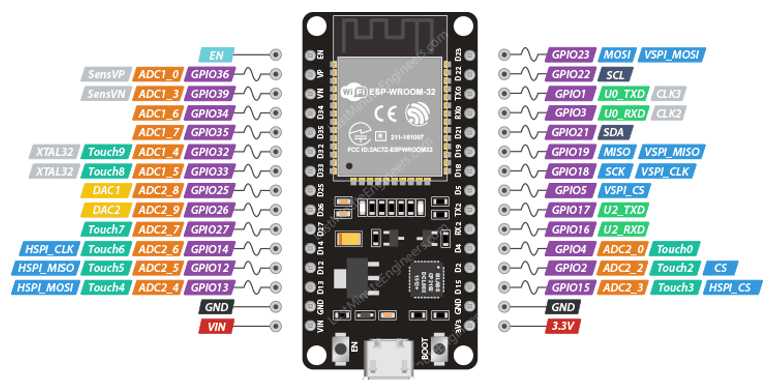
Figure : Block

### **4.1.1 Control block**

* Function: Allows motor control
* Used:
* Specifications: The same 2.4

### **4.1.2 Processing Block**

* Function: Processing signals sent to sensors and control block
* Used: ESP32 Devkit
* Specifications: The same 2.1
* *Pins connected to* ESP32 Devkit

****

**L298 DC Motor Driver:**

* 5V connect to 5V source and 5V of Esp32
* GND connect to GND of Esp32 and negative source of battery power
* 12V connect to battery positive power
* PIN OUT (1-4) connect in sequence with 4 motors

**Servo motor SG90:**

* Vcc connect to 5V source
* GND connect to GND of Esp32

### **4.1.3 Sensor Block**

**Ultrasonic sensor module SRF05:x**

* Vcc connect to 5V source
* GND connect to GND of Esp32
* The pins to transmit trin and echo data are 19 ,18 pins of esp32

## **4.2 Model Design**

### **4.2.1 Design APP on MIT inventor**

A screen shot of a tablet

Description automatically generated with medium confidence

Figure : WORKING DISPLAY 1

A screen shot of a device

Description automatically generated with low confidence

Figure :WORKING DISPLAY 2

### **4.2.2. Logical connection diagram**

A screenshot of a computer

Description automatically generated with medium confidence

Figure :DIAGRAM OF VOICE CONTROL

*A screenshot of a computer screen

Description automatically generated with low confidence*

Figure :DIAGRAM OF REMOTE CONTROL

### **4.2.3 App in mobie phone**

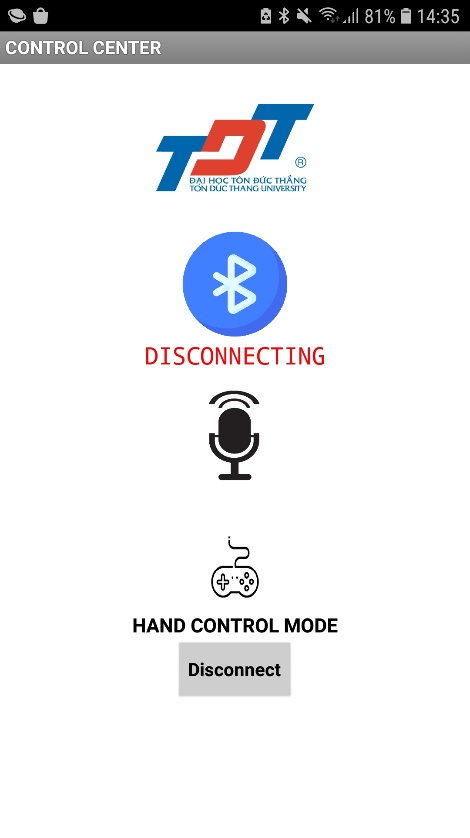


Figure : Screen 1 in phone

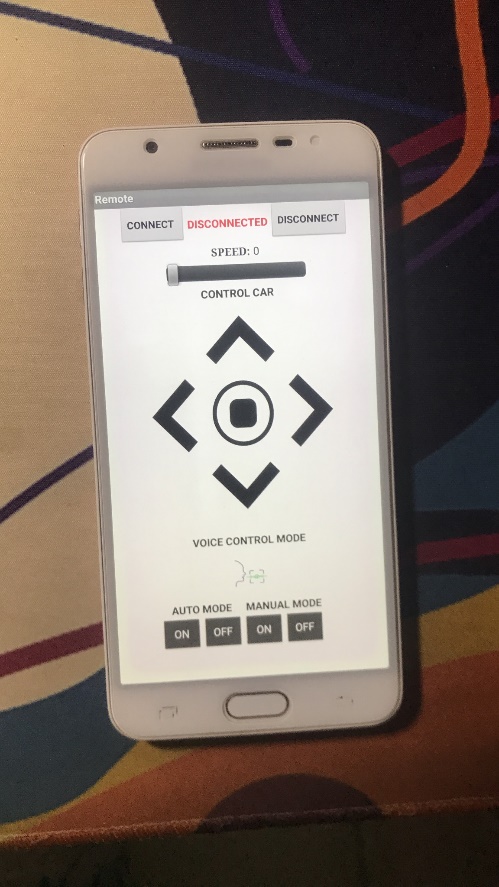


Figure : Screen 2 in phone

## **4.3 Experiment**

### **4.3.1 Implementation process**

* Step 1: Turn on switch to start the systems.
* Step 2: Open the MIT app.
* Step 3: Connect phone to esp32 via Bluetooth.
* Step 4: Control car

### **4.3.2 Results**

* Works as expected.
* The engines run as desired.
* Meet the requirements as set out at the beginning.

### **4.3.3 Experiment overview**

Need to improve Bluetooth stability during control, after this test i learned how to use and design phone app. Understand how the motor works and the principle of the PMW wave

# **CHAPTER 5. CONCLUSION**

## **5.1 Advantage**

- Simple circuit, easy to use, can adjust two desired modes

- Friendly user

## **5.2 Disadvantage**

**-** The application of the MIT app has not been fully exploited yet

- The connection is not stable

## **5.3 Development**

- Improved system stability (PID)

- Will learn to improve combined with how the robot hand uses 4 servo.

# **REFERENCES**

1. Wikipedia contributors. "Pulse-width modulation." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 27 Sep. 2022. Web. 8 Nov. 2022.
2. Wikipedia contributors. (2023, May 4). ESP32. In *Wikipedia, The Free Encyclopedia*.
3. Lear MIT app at [Troubleshooting for App Inventor 2 (mit.edu)](http://appinventor.mit.edu/explore/ai2/support/troubleshooting) and youtube.com
4. ["Relay H-bridge (Relay Motor controller)"](http://nvhs.wordpress.com/project/catspberry/motor-controller/). 11 December 2012.
5. Al Williams (2002). [Microcontroller projects using the Basic Stamp](https://books.google.com/books?id=E0_07GfXo2wC&pg=PA344) (2nd ed.). Focal Press. p. 344. [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [978-1-57820-101-3](https://en.wikipedia.org/wiki/Special:BookSources/978-1-57820-101-3).
6. *Markus Ulsass (2015-12-25).*["ESP32 beta module HiRes pictures"](http://lookmanowire.blogspot.com/2015/12/esp32-beta-module-hi-res-pictures.html). Retrieved 2016-09-02*.*
7. [Limor Fried](https://en.wikipedia.org/wiki/Limor_Fried" \o "Limor Fried)*(2015-12-22).*[Playing With New ESP32 Beta Module](https://www.youtube.com/watch?v=HCGHb0OVz1s)*. Adafruit Industries*. Retrieved 2016-09-02*.*
8. *Jim Lindblom (2016-01-21).*["Enginursday: First Impressions of the ESP32"](https://www.sparkfun.com/news/2017)*. Sparkfun Electronics*. Retrieved 2016-09-01*.*

# **APPENDIX 1**

#include <BluetoothSerial.h>

const int trigPin = 19 ;

const int echoPin = 18;

const int enablePinA = 33; // Điều khiển tốc độ động cơ bên trái GPIO5(D1)

const int motorPin1 = 25; // L298N in1 Động cơ trái quay GPIO4(D2)

const int motorPin2 = 26; // L298N in2 Động cơ trái quay ngược lại GPIO0(D3)

const int motorPin3 = 27; // L298N in3 Động cơ phải quay GPIO2(D4)

const int motorPin4 = 14;

const int enablePinB = 12; // Điều khiển tốc độ động cơ bên phải GPIO12(D6)

const int turnDelay = 1000;

const int stopDelay = 300;

const int minDistance = 25;

int status1= false ;

int status2= false ;

int motorSpeed ;

long khoangcach1 = 0;

char kytu ;

String chuoi;

int value ;

// Khai báo biến PWM

const int freq = 30000;

const int channel = 0;

const int resolution = 8;

// Khai báo biến bluetooth

BluetoothSerial serialBT;

void setup()

{

// put your setup code here, to run once:

Serial.begin(9600);

serialBT.begin("Control Car");

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(enablePinA, OUTPUT);

pinMode(motorPin1, OUTPUT);

pinMode(motorPin2, OUTPUT);

pinMode(motorPin3, OUTPUT);

pinMode(motorPin4, OUTPUT);

pinMode(enablePinB, OUTPUT);

// Khởi tạo PWM để điều khiển tốc độ động cơ

ledcSetup(channel, freq, resolution);

ledcAttachPin(enablePinA, channel);

ledcAttachPin(enablePinB, channel);

// ttc=0; //trang thai nut nhan chuyen

}

void loop()

{

voice();

if (serialBT.available())

{

//value = serialBT.parseInt();

value = Serial.read();

motorSpeed = map(value, 0, 255, 0, 255);

ledcWrite(channel, motorSpeed);

Serial.print("Motor speed: ");

Serial.println(value);

kytu=serialBT.read();

chuoi=chuoi+kytu;

Serial.println(kytu);

if (kytu == 'A')

{

status1 = true;

}

else if (kytu == 'T')

{

status1 = false;

}

if (kytu == 'M')

{

status2 = true;

}

else if (kytu == 'N')

{

status2 = false;

}

}

if (status1)

{

autocar();

}

if (status2)

{

manual();

}

}

void voice()

{

if(chuoi.indexOf("thẳng")>=0)

{

carforward();

delay(2000);

carstop();

chuoi="";

}

if(chuoi.indexOf("lùi")>=0)

{

carbackward();

delay(2000);

carstop();

Serial.println(chuoi);

chuoi="";

}

if(chuoi.indexOf("trái")>=0)

{

carturnleft();

delay(500);

carstop();

Serial.println(chuoi);

chuoi="";

}

if(chuoi.indexOf("phải")>=0)

{

carturnright();

delay(500);

carstop();

Serial.println(chuoi);

chuoi="";

}

}

void manual()

{

Serial.write(kytu);

switch(kytu)

{

case 'U':

carforward();

break;

case 'D':

carbackward();

break;

case 'L':

carturnleft();

break;

case 'R':

carturnright();

break;

case 'S':

carstop();

break;

}

}

void carforward()

{

digitalWrite(motorPin1, HIGH);

digitalWrite(motorPin2, LOW);

digitalWrite(motorPin3, HIGH);

digitalWrite(motorPin4, LOW);

ledcWrite(channel, motorSpeed);

}

void carbackward() {

digitalWrite(motorPin1, LOW);

digitalWrite(motorPin2, HIGH);

digitalWrite(motorPin3, LOW);

digitalWrite(motorPin4, HIGH);

ledcWrite(channel, motorSpeed);

}

void carturnleft()

{

digitalWrite(motorPin1, HIGH);

digitalWrite(motorPin2, LOW);

digitalWrite(motorPin3, LOW);

digitalWrite(motorPin4, HIGH);

ledcWrite(channel, motorSpeed);

}

void carturnright()

{

digitalWrite(motorPin1, LOW);

digitalWrite(motorPin2, HIGH);

digitalWrite(motorPin3, HIGH);

digitalWrite(motorPin4, LOW);

ledcWrite(channel, motorSpeed);

}

void carstop() {

digitalWrite(motorPin1, LOW);

digitalWrite(motorPin2, LOW);

digitalWrite(motorPin3, LOW);

digitalWrite(motorPin4, LOW);

}

void autocar() {

long khoangcach = 0;

khoangcach = dokhoangcach();

if (khoangcach > minDistance || khoangcach == 0) {

khoangcach = dokhoangcach();

if (khoangcach > minDistance || khoangcach == 0) {

carforward();

Serial.println("Di thang");

}

}

else {

carstop();

delay(stopDelay);

khoangcach1 = dokhoangcach();

if (khoangcach1 < 15) {

carstop();

carbackward();

delay(turnDelay);

carstop();

delay(stopDelay);

carturnleft();

delay(turnDelay);

carstop();

delay(stopDelay);

}

else {

carstop();

carforward();

Serial.println("Di thang");

}

}

}

long dokhoangcach() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

long thoigian = pulseIn(echoPin, HIGH);

long khoangcach = thoigian / 2 / 29.412;

return khoangcach;

}