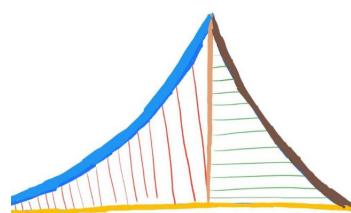

EMBEDDED SYSTEM

G. V. V. Sharma



Copyright ©2023 by G. V. V. Sharma.

<https://creativecommons.org/licenses/by-sa/3.0/>

and

<https://www.gnu.org/licenses/fdl-1.3.en.html>

Contents

Introduction	iii
1 Vaman-ESP32	1
1.1 Flash Vaman-ESP32 using Arduino	1
1.2 Measuring Unknown Resistance Using Vaman-ESP	4
1.2.1 Components	4
1.2.2 Setting up the Display	5
1.2.3 Measuring the resistance	6
1.2.4 Displaying the Measured resistance on LCD and website	8
1.2.5 Explanation	9
1.3 I2C Communication Between Vaman-ESP and Arduino	10
1.3.1 Components	11
1.3.2 Setting up the Master and Slave	11
1.4 UART Communication between Vaman-ESP and Arduino	12
1.4.1 Components	12
1.4.2 Connections	12
1.4.3 Measuring the resistance	13
1.4.4 Displaying the Measured resistance on website	14
1.5 Bluetooth Controlled Seven segment Display	15

1.5.1 Components	15
1.5.2 Connections	15
2 Toy car	19
2.1 Components Table	19
2.1.1 (Vaman Board ESP32)	19
2.1.2 DC Motors	20
2.1.3 L293 motor driver	20
2.1.4 USB UART	21
2.1.5 UGV frame/chassis	21
2.1.6 Batteries for powering various equipments	22
2.2 Assembling the UGV kit	22
2.3 Circuit Connections	23
2.3.1 Working	25
2.4 Code Execution For Wifi Toycar	26
2.5 Code Execution for Bluetooth Toycar	27

Introduction

This book introduces Embedded Systems through using the Vaman framework.

Chapter 1

Vaman-ESP32

1.1. Flash Vaman-ESP32 using Arduino

1.1.1. Make sure that Vaman board do not power any devices.

1.1.2. Make connections as shown in Table 1.1.3.1 and Fig. 1.1.3.1.

1.1.3. The Vaman pin diagram is available in Fig. 1.1.3.2

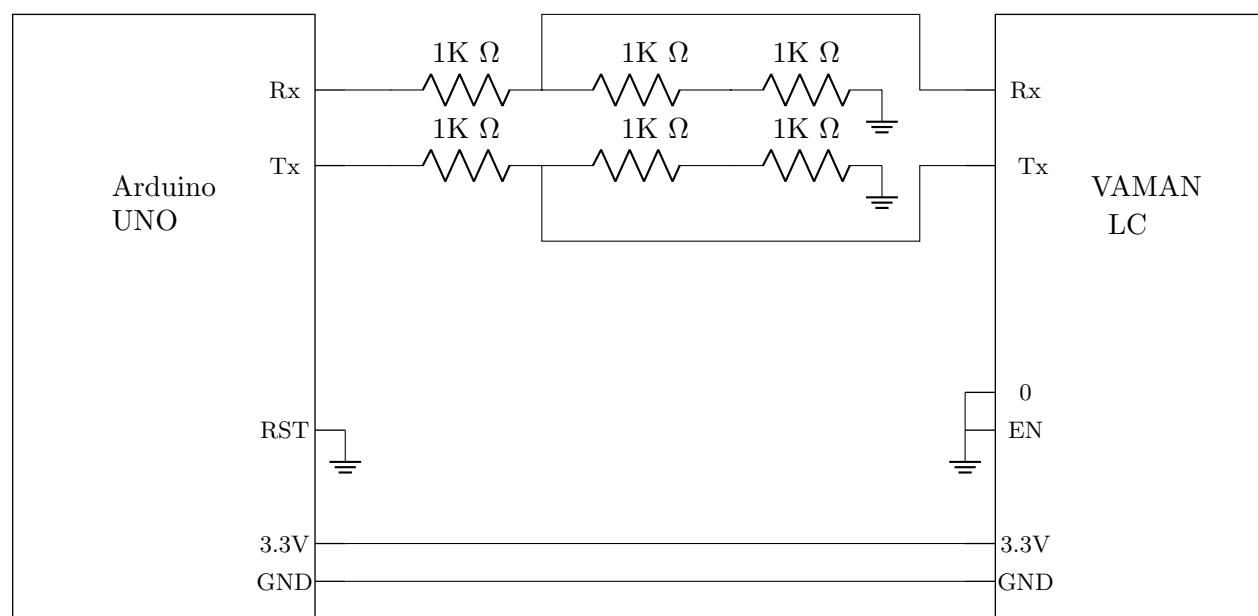


Figure 1.1.3.1: Circuit Connections

VAMAN LC-1

PINOUT

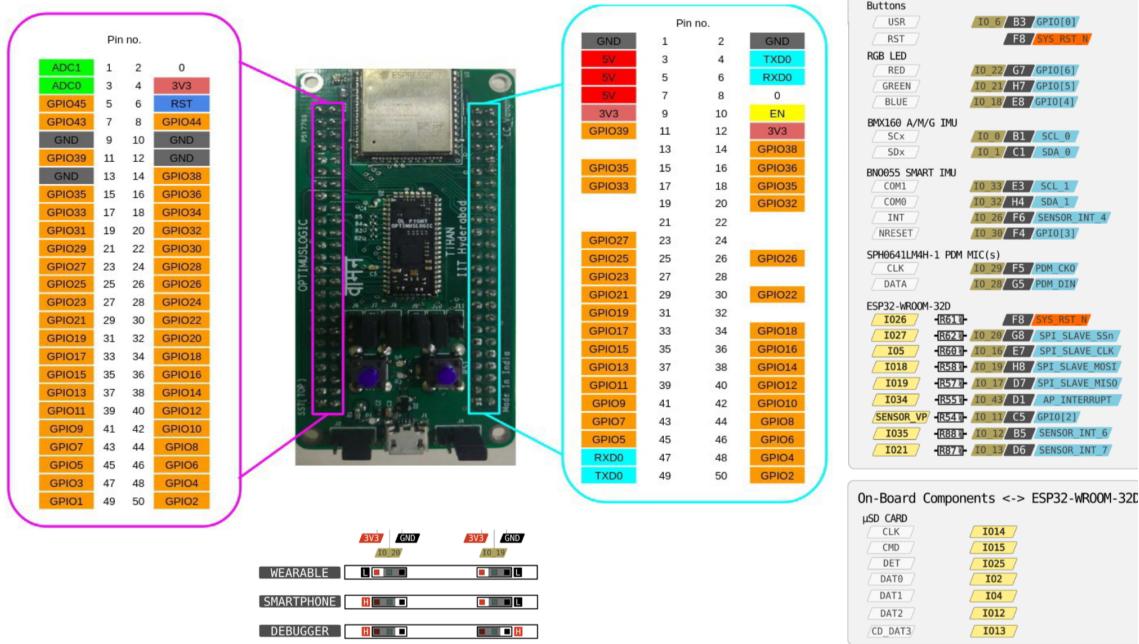


Figure 1.1.3.2: Vaman pins

VAMAN LC PINS	ARDUINO PINS
3.3	3.3
GND	GND
TXD0	TXD
RXD0	RXD
0	GND
EN	GND

Table 1.1.3.1:

1.1.4. For compiling and generating the bin file

1.1.5. make sure that platformio.ini file contains these lines

```
[env:esp32doit-devkit-v1]
platform = espressif32
board = esp32doit-devkit-v1
framework = arduino
platform_packages = toolchain-xtensa-esp32@https://github.com/esphome/
    esphome-docker-base/releases/download/v1.4.0/toolchain-xtensa32.tar.gz
framework-arduinoespressif32@<3.10006.210326
```

1.1.6. For uploading bin file to Vaman through ArduinoDroid application

1. Open the Droid Application
2. Click the three dots **in** the top right corner
3. Navigate to Settings → Board Type
4. Select ESP32 → DOIT ESP32 DEVKIT V1
5. Change the upload speed to 115200
6. Upload the generated .bin file

1.1.7. While the dots are printed on the screen, disconnect the EN wire from GND. Make sure that the Vaman board is not powering any device while flashing. The Vaman-ESP should now flash.

1.1.8. After flashing, disconnect pin 0 on Vaman-ESP from GND. Power on Vaman appropriately.

1.2. Measuring Unknown Resistance Using Vaman- ESP

Through this manual, we learn how to measure an unknown resistance through Vaman-ESP and display it on an LCD.

1.2.1. Components

Component	Value	Quantity
Resistor	220 Ohm	1
	1K	1
ESP32	Devkit V1	1
Jumper Wires		20
Bread board		1
LCD	16 X 2	1
Potentiometer	10K	1

Table 1.2.1: Components

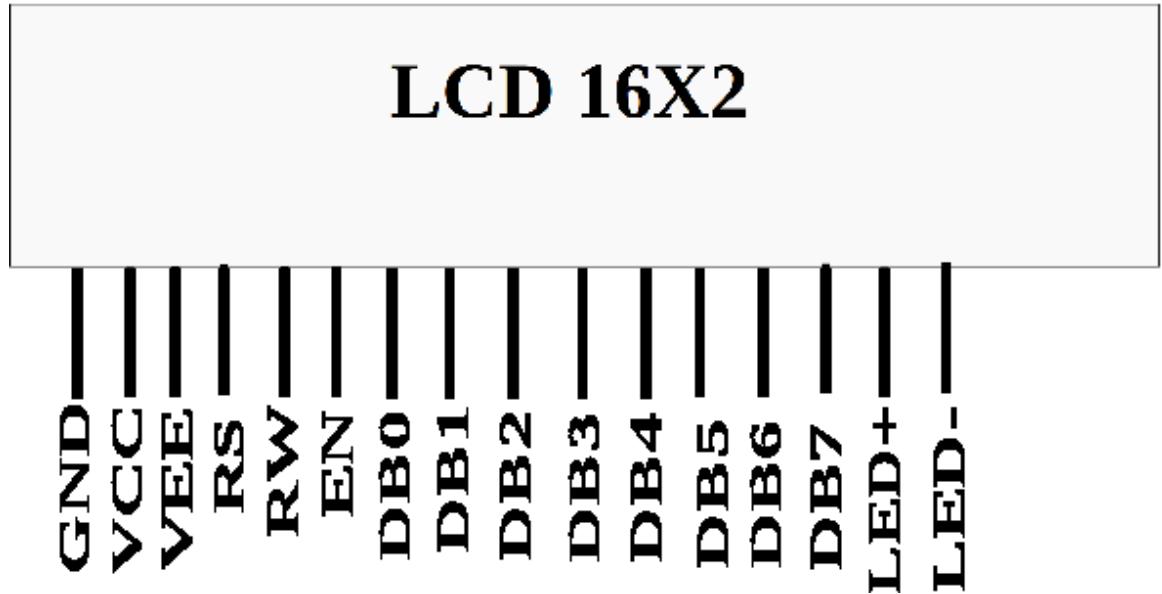


Figure 1.2.1.1: LCD pins

1.2.2. Setting up the Display

1.2.1. Plug the LCD in Fig. 1.2.1.1 to the breadboard.

1.2.2. Connect the Vaman-ESP pins to LCD pins as per Table 1.2.3.1. Make sure that all 5V sources are connected to the LCD through a $220\ \Omega$ resistance.

1.2.3. The Vaman pin diagram is available in Fig. 1.1.3.2

1.2.4. Execute the following code after editing the wifi credentials

```
vaman/vaman-esp/lcd/codes/setup
```

You should see the following message

```
Hi
```

```
This is CSP Lab
```

ESP32	LCD Pins	LCD Pin Label	LCD Pin Description
GND	1	GND	
5V	2	Vcc	
GND	3	Vee	Contrast
GPIO 19	4	RS	Register Select
GND	5	R/W	Read/Write
GPIO 23	6	EN	Enable
GPIO 18	11	DB4	Serial Connection
GPIO 17	12	DB5	Serial Connection
GPIO 16	13	DB6	Serial Connection
GPIO 15	14	DB7	Serial Connection
5V	15	LED+	Backlight
GND	16	LED-	Backlight

Table 1.2.3.1: Make sure that all 5V sources are connected to the LCD through a 220Ω resistance.

1.2.5. Modify the above code to display your name.

1.2.3. Measuring the resistance

1.2.1. Connect the 5V pin of the Vaman-ESP to an extreme pin of the Breadboard shown in Fig. 1.2.1.1. Let this pin be V_{cc} .

1.2.2. Connect the GND pin of the Vaman-ESP to the opposite extreme pin of the Breadboard.

1.2.3. Let R_1 be the known resistor and R_2 be the unknown resistor. Connect R_1 and R_2

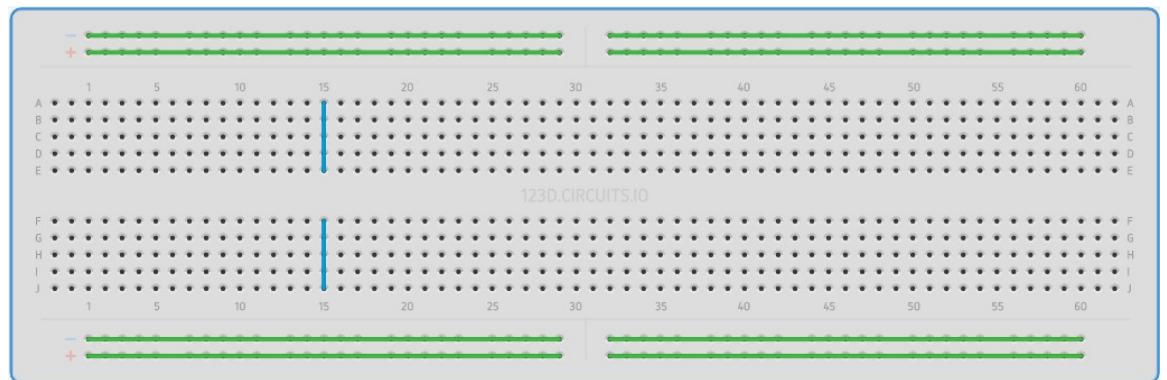


Figure 1.2.1.1: Breadboard

in series such that R_1 is connected to V_{cc} and R_2 is connected to GND. Refer to Fig. 1.2.3.1

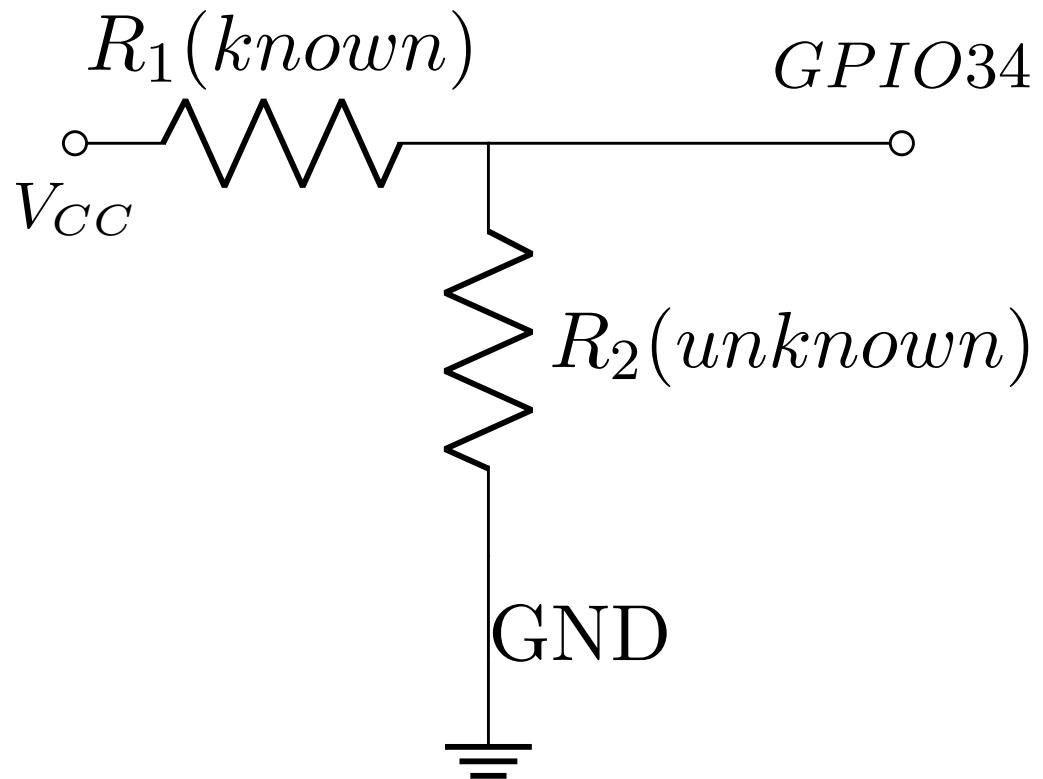


Figure 1.2.3.1: Voltage Divider

1.2.4. Connect the junction between the two resistors to the GPIO34 pin on the Vaman-ESP.

1.2.5. Connect the Vaman-ESP to the computer so that it is powered.

1.2.6. Execute the following code after editing the wifi credentials

```
vaman/vaman-esp/lcd/codes/resistance
```

1.2.4. Displaying the Measured resistance on LCD and website

1.2.1. The unknown resistance is measured and displayed the measured resistance on the LCD display and also on the Vaman-ESP webserver.

1.2.2. Connect the Vaman-ESP pins to LCD pins as per Table 1.2.3.1.

1.2.3. Execute the following code after editing the wifi credentials

```
vaman/vaman-esp/lcd/webserver/codes
```

1.2.4. After flashing the code to vaman-ESP, the board will be connected to the wifi credentials provided.

1.2.5. Now connect the same WiFi credentials to the mobile phone for accessing the IP address, which can be accessed by

```
ifconfig  
nmap -sn 192.168.x.x/24
```

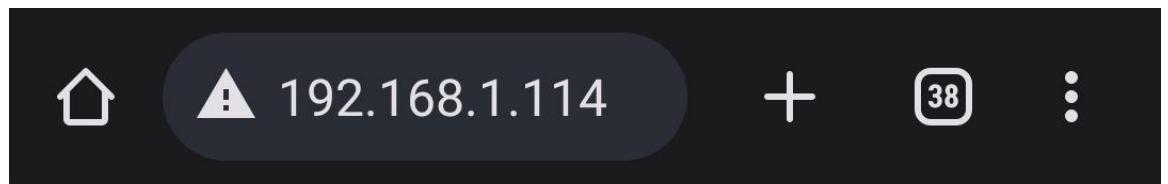
1.2.6. Change the IP address in the second command accordingly with the IP address provided by first command.

1.2.7. By the above commands the IP address of vaman-ESP will be displayed.

1.2.8. Now the vaman-ESP will be hosting a webserver

1.2.9. Inorder to access the webserver type the IP address of the vaman-ESP in the web browser.

1.2.10. In the website loaded by the IP address of vaman-ESP the Unknown resistance is displayed as shown in Fig. 1.2.10.1



Measured Resistance: 190.75 Ohms

Figure 1.2.10.1: Website

1.2.5. Explanation

1.2.1. We create a variable called analogPin and assign it to 0. This is because the voltage value we are going to read is connected to analogPin GPIO34.

1.2.2. The 12-bit ADC can differentiate 4096 discrete voltage levels, 5 volt is applied to 2 resistors and the voltage sample is taken in between the resistors. The value which we get from analogPin can be between 0 and 4095. 0 would represent 0 volts falls across the unknown resistor. A value of 4095 would mean that practically all 5 volts falls across the unknown resistor.

1.2.3. V_{out} represents the divided voltage that falls across the unknown resistor.

1.2.4. The Ohm meter in this manual works on the principle of the voltage divider shown in Fig. 1.2.3.1.

$$V_{out} = \frac{R_1}{R_1 + R_2} V_{in} \quad (1.2.4.1)$$

$$\Rightarrow R_2 = R_1 \left(\frac{V_{in}}{V_{out}} - 1 \right) \quad (1.2.4.2)$$

In the above, $V_{in} = 5V$, $R_1 = 220\Omega$.

1.2.5. Repeat the exercise with another unknown resistance.

1.3. I2C Communication Between Vaman-ESP and Arduino

Through this manual, we will learn how to setting up the vaman-ESP as a Master and Arduino as a Slave using I2C protocol.

Component	Value	Quantity
ESP32	Devkit V1	1
Arduino	UNO	1
Connecting Wires		30
LCD	16 X 2	1

Table 1.3.1: Components

1.3.1. Components

1.3.2. Setting up the Master and Slave

1.3.1. Connect the vaman-ESP pins to Arduino pins as per Table 1.3.1.1.

I2C	ESP32	Arduino
SDA	GPIO 21	A4
SDC	GPIO 22	A5
	VCC	VCC
	GND	GND

Table 1.3.1.1:

1.3.2. Connect the vaman-ESP pins to LCD pins as per 1.2.3.1..

1.3.3. The Vaman pin diagram is available in Fig. 1.1.3.2

1.3.4. Configure Arduino Uno as a Slave using the following code.

```
vaman/vaman-esp/I2C/codes/I2C_Sender_Arduino/src/main.cpp
```

1.3.5. Now configure vaman-ESP as a Master using the following code.

```
vaman/vaman-esp/I2C/codes/I2C_Reciever_ESP32/src/main.cpp
```

1.4. UART Communication between Vaman-ESP and Arduino

Through this manual, we learn how to communicate between Vaman-ESP32 and Arduino UNO through UART Protocol. The Unknown resistance is measured using Arduino and sending the value to Vaman through UART and displaying the unknown Resistance on ESP-Webserver.

1.4.1. Components

Component	Value	Quantity
Resistor	220 Ohm	1
	1K	1
Vaman	LC	1
Arduino	UNO	1
Jumper Wires		10
Bread board		1

Table 1.4.2: Components

1.4.2. Connections

1.4.1. Connect the Vaman and Arduino as shown Table. 1.4.1.2.

Arduino UNO	Vaman-ESP
Rx(Pin-0)	17 (Tx)
Tx (Pin-1)	16 (Rx)

Table 1.4.1.2: Connections

1.4.2. The Vaman pin diagram is available in Fig. 1.1.3.2

1.4.3. Upload the following code to Arduino UNO

```
vaman/vaman-esp/UART/codes/UNO
```

1.4.3. Measuring the resistance

1.4.1. Connect the 5V pin of the Vaman-ESP to an extreme pin of the Breadboard shown in Fig. 1.2.1.1. Let this pin be V_{cc} .

1.4.2. Connect the GND pin of the Vaman-ESP to the opposite extreme pin of the Breadboard.

1.4.3. Let R_1 be the known resistor and R_2 be the unknown resistor. Connect R_1 and R_2 in series such that R_1 is connected to V_{cc} and R_2 is connected to GND. Refer to Fig. 1.2.3.1

1.4.4. Connect the junction between the two resistors to the A0 pin on the Arduino board.

1.4.5. Now Power the Vaman board

1.4.6. Execute the following code after editing the wifi credentials

```
vaman/vaman-esp/UART/codes/VAMAN
```

1.4.4. Displaying the Measured resistance on website

1.4.1. The unknown resistance is measured and displayed the measured resistance on the Vaman-ESP webserver.

1.4.2. After flashing the code to vaman-ESP, the board will be connected to the wifi credentials provided.

1.4.3. Now connect the same WiFi credentials to the mobile phone for accessing the IP address, which can be accessed by

```
ifconfig  
nmap -sn 192.168.x.x/24
```

1.4.4. Change the IP address in the second command accordingly with the IP address provided by first command.

1.4.5. By the above commands the IP address of vaman-ESP will be displayed.

1.4.6. Now the vaman-ESP will be hosting a webserver

1.4.7. Inorder to access the webserver type the IP address of the vaman-ESP in the web browser.

1.4.8. In the website loaded by the IP address of vaman-ESP the Unknown resistance is displayed as shown in Fig. 1.2.10.1

1.5. Bluetooth Controlled Seven segment Display

play

This manual shows how to control the Seven Segment Display through the Dabble android application using Bluetooth in Digital mode and display on the seven segment according to the controls in the android app.

1.5.1. Components

Component		Quantity
Resistor	220 Ohm	1
Seven Segment Display		1
Vaman	LC	1
Arduino	UNO	1
Jumper Wires		10
Bread board		1

Table 1.5.2: Components

1.5.2. Connections

1.5.1. Connect the Arduino-UART to VAMAN as per Table. 1.1.3.1 and Figure 1.1.3.1.

1.5.2. Now, execute the following code

```
vaman/vaman-esp/bluetooth/codes/src
```

1.5.3. Make sure to give the path to DabbleESP32-master folder path in the platformio.ini file as shown below

```
lib_extra_dirs = /"Path to DabbleESP32-master folder"/DabbleESP32/src
```

- 1.5.4. Install the Dabble Android application and give the necessary permissions.
- 1.5.5. Connect the bluetooth of vaman ESP-32 bluetooth to the mobile- where the bluetooth is labled as "ESP-32"
- 1.5.6. Open the Dabble application. Select gamepad option in the app and then select Digital Mode and connect it app to ESP-32 by connecting it ESP-32 bluetooth as shown in Figure 1.5.6.1.

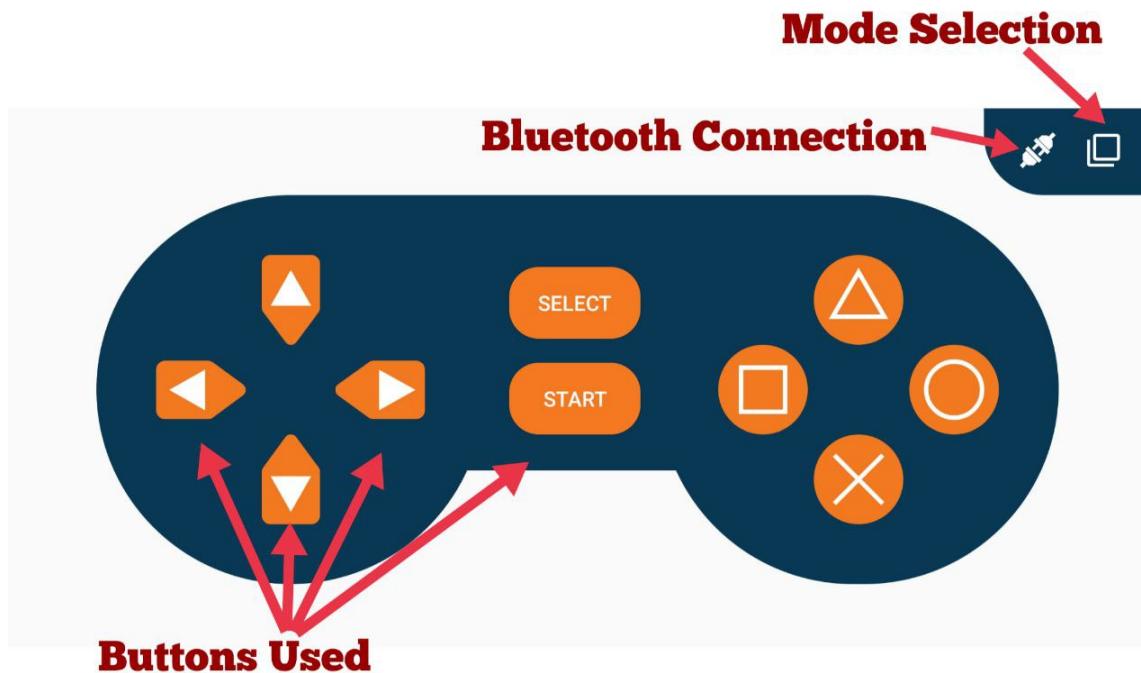


Figure 1.5.6.1: Dabble app Interface

- 1.5.7. Now connect the Seven Segment to the Vaman board according to the given Table.

1.5.7.2

VAMAN ESP pins	Seven Segment pins
IO-32	a
IO-33	b
IO-25	c
IO-26	d
IO-27	e
IO-14	f
IO-12	g

Table 1.5.7.2: Connections

1.5.8. Now you can observe the changes on sevensegment display for Start, Up, Down, Right and Left keys pressed on the Digital Mode on the android application

Chapter 2

Toy car

: Through this manual, we learn how to communicate between SPI, Wishbone Interfacing and Address Mapping. On the Vaman Board, we have an EOS S3 and ESP32. The Communication between these two happens via SPI i.e, Serial Peripheral Interface. And this is facilitated only when all the 4 jumpers on the board are closed.

2.1. Components Table

Components	Quantity
Vaman Board ESP32	1
USB UART	1
UGV Chasis	1
L293 Motor Driver	1
DC Motors	2
Jumper wires	15
Bread board	1

Table 2.1.0.2: components table of toycar

2.1.1. (Vaman Board ESP32)

2.1.1. Refer the Vaman Board ESP32 Fig. 1.1.3.2

2.1.2. DC Motors



Figure 2.1.1.1: DC motors

2.1.3. L293 motor driver



Figure 2.1.1.2: L293 motor driver

2.1.4. USB UART

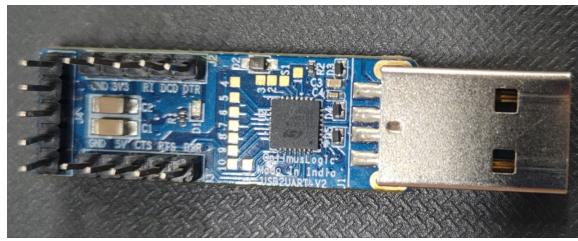


Figure 2.1.1.3: USB UART

2.1.5. UGV frame/chassis



Figure 2.1.1.4: UGV frame/chassis

2.1.6. Batteries for powering various equipments

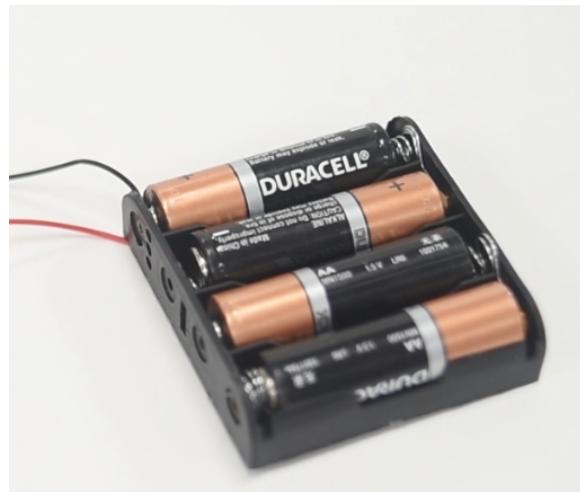


Figure 2.1.1.5: Batteries for powering various equipments

2.2. Assembling the UGV kit

2.2.2. Assemble the Chassis using the provided nuts/screws, Wheels, and parts.



Figure 2.2.2.1: screws connecting

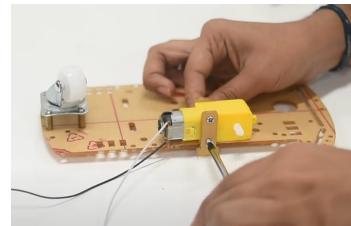


Figure 2.2.2.2: Dc motors connecting



Figure 2.2.2.3: wheels connections

2.3. Circuit Connections

2.3.3. Make the Circuit Connections as per the Table 2.3.3.2.

Vaman Board	Motor Driver Unit
Pin 21	Right Motor Input 1
Pin 18	Right Motor Input 2
Pin 23	Left Motor Input 1
Pin 22	Left Motor Input 2
5v	VCC
GND	GND

Table 2.3.3.2: connection with vaman board

2.3.4. Make the Motor Driver Connections as per the Table 2.3.4.2.

INPUT	VAMAN BOARD	OUTPUT	MOTOR
A1	PYGMY 21	Vcc	5V
A2	PYGMY 18	GND	GND
EN	-	MA1	MOTOR A1
VCC	5V	MA2	MOTOR A2
B2	PYGMY 23	MB1	MOTOR B1
B1	PYGMY 22	MB2	MOTOR B2
5V	VCC	-	-
GND	GND	-	-

Table 2.3.4.2: connection with L293 Motor Driver

Vaman Board ESP 32	Motor Driver Unit
Pin 16	Right Motor Input 1
Pin 17	Right Motor Input 2
Pin 18	Left Motor Input 1
Pin 19	Left Motor Input 2
5v	VCC
GND	GND

Table 2.3.5.2: WIFI CAR Connections

2.3.5. Make the Circuit Connections as per the Table 2.3.5.2.

2.3.6. Connect the USB-UART pins to the Vaman ESP32 pins according to Table 2.3.6.2.

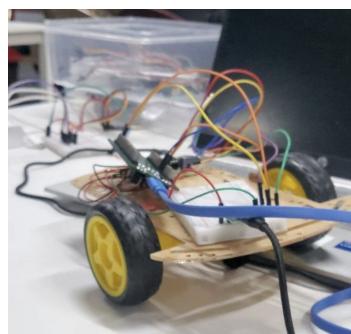


Figure 2.3.6.1: After all connections

Vaman LC PINS	UART PINS
GND	GND
ENB	ENB
TXD0	RXD
RXDO	TXD
0	IO 0
5V	5V

Table 2.3.6.2: USB UART Connections

- 2.3.7. Download the “dabble” application from the play store on an Android phone.
- 2.3.8. Using dabble application, connect to the ESP32 on the UGV kit using Bluetooth connection.
- 2.3.9. Control the navigation of the UGV kit using the GUI controls on the dabble application.

2.3.1. Working

On the hardware level there are three key points: SPI,Wishbone Interfacing and Address Mapping. Vaman Board, we have an EOS S3 and ESP32. The Communication between these two happens via Serial Peripheral Interface(SPI).

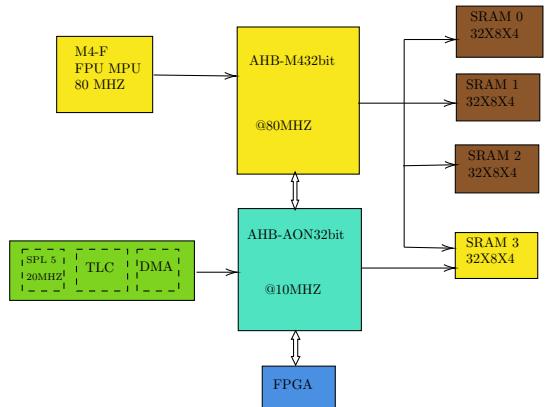


Figure 2.3.9.1: EOS S3 Architecture

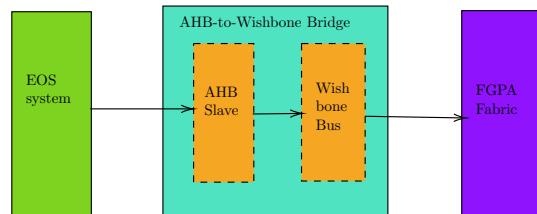


Figure 2.3.9.2: Wishbone Slave Interface

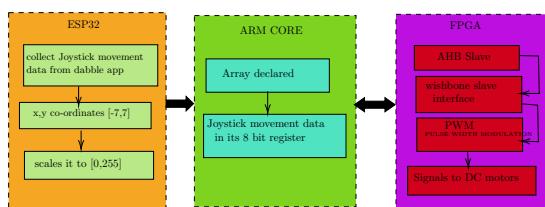


Figure 2.3.9.3: Hardware Block level Architecture

2.4. Code Execution For Wifi ToyCar

2.4.10. Now, Execute the following code

```
vaman/toycar/codes/wifi_toycar/src
```

2.4.11. Build the ESP32 firmware

```
cd wifi_toycar  
pio run
```

2.4.12. Flash ESP32 firmware (connect USB-UART adapter)

```
pio run -t upload
```

2.4.13. Connect your own TAB /Phone Hot spot and Enter Your SSID and Password

```
const char* ssid = "fwc"; /*Enter Your SSID*/  
const char* password = "fwc123"; /*Enter Your Password*/
```

2.4.14. Install the **Dabble** app on the Mobile from the **Playstore**. Connect it to the **ESP32** on the Vaman Board using **wifi**. Change the controls to **Joystick mode** to navigate the UGV.

2.5. Code Execution for Bluetooth Toycar

2.5.15. Now, Execute the following code

```
vaman/toycar/codes/bluetooth_toycar
```

2.5.16. Build the ESP32 firmware

```
cd esp32_pwmctrl  
pio run
```

2.5.17. Flash ESP32 firmware (connect USB-UART adapter)

```
pio run -t nobuild -t upload
```

2.5.18. If using termux, send .pio/build/esp32doit-devkit-v1/firmware.bin to PC using

```
scp .pio/build/esp32doit-devkit-v1/firmware.bin Username@IPAddress:
```

2.5.19. Modify line 140 of config.mk to setup path to pygmy-sdk and then Build m4 firmware using

```
cd m4_pwmctrl/GCC_Project  
make
```

2.5.20. If using termux, send output/m4_pwmctrl.bin to PC using

```
scp output/m4_pwmctrl.bin username@IPAddress:
```

2.5.21. Build fpga source (.bin file)

```
cd fpga_pwmctrl/rtl  
ql_symbiflow -compile -d ql-eos-s3 -P pu64 -v *.v -t AL4S3B_FPGA_Top -p  
quickfeather.pcf -dump jlink binary
```

2.5.22. If using termux, send AL4S3B_FPGA_Top.bin to PC using

```
scp AL4S3B_FPGA_Top.bin username@IPAddress:
```

2.5.23. Connect usb cable to vaman board and Flash eos s3 soc, using

```
sudo python3 <Type path to tiny fpga programmer application> --port /dev/  
ttyACM0 --appfpga AL4S3B_FPGA_Top.bin --m4app m4_pwmctrl.bin --  
mode m4-fpga --reset
```

2.5.24. Install the **Dabble app** on the Mobile from the **Playstore**. Connect it to the **ESP32** on the Vaman Board using **Bluetooth**. Change the controls to **Joystick mode** to navigate the UGV.

