In order to use the WPI300 keypad the library had to be installed on Arduino IDE.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

The keypad library was downloaded and enabled as shown which ‘includes’ the relevant libraries at the top of the code. This will allow commands specific to the keypad to be used.

A picture containing text, orange

Description automatically generated

Graphical user interface, application

Description automatically generatedAlso allows access to the example codes for the keypad as shown.

Pin layout according to the data sheet.

<https://esp32io.com/tutorials/esp32-keypad> (Used to determine which pins the keypad should be attached to.

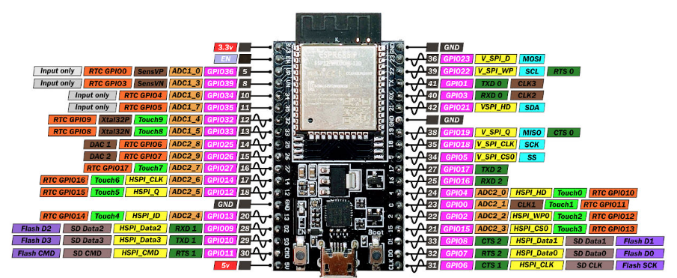
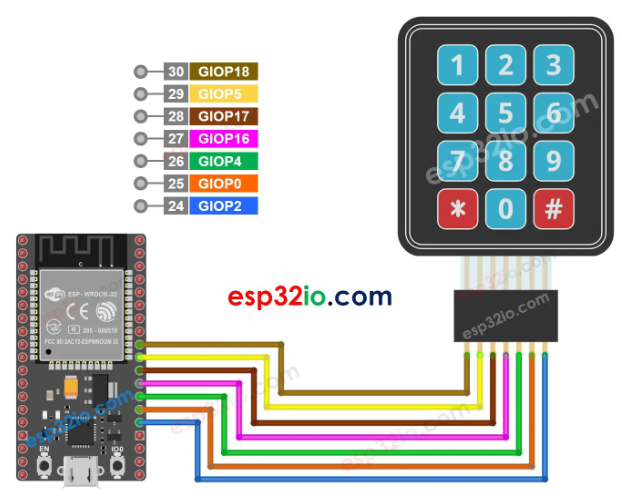


Figure 16: Pinout diagram of ESP32 microcontroller [12]



The ESP used for this diagram is a different model but using the pin out diagram for our ESP32 as above, the pins were connected as shown in photo.



Changed the pin numbers in the example code to the numbers as above so that the data was expected to be received from the correct pins. In the above code the pin names are slightly different so only the numbers have been copied, the original variable names from the example code remain the same. 1st screenshot is original example, 2nd is the updated code.

A picture containing graphical user interface

Description automatically generated



When uploaded this code partially worked. When the serial monitor was opened a number was displayed after each number was pressed. The first column was correct, however, column 2 and 3 seemed to be swapped eg. When 2 is pressed, 3 is displayed etc. Should be fixed by swapping the pin numbers in the code or by swapping the order that the wires are2 attached. Decided to change the code as the wires were neatly in order and swapping them would mean making them cross over which is not as aesthetically pleasing. Screen shot below is original outputs displaying the wrong numbers.

Graphical user interface, text, application

Description automatically generated

Screenshot below shows updated code with the column pins swapped, that line of code is highlighted, and serial monitor showing values pressed in order. This means that the value displayed is now accurate and matches the button pressed.

Graphical user interface, text, application

Description automatically generated

Code saved as ‘Keypad displaying number pressed’.

Actually did have 2 wires crossed over, the red and white, these were swapped and the pin numbers in the code were swapped back.

The wires were attached to a header to make plugging in all of them easier, and this was then touched to the ESP32 that was plugged into the EEEBot. When this was done the previously function display code no longer worked. To try to identify the problem the ESP32 was returned to the breadboard and the wires placed into the header then tested to see if the code worked.

Unfortunately, it still didn’t work. As the legs to the headers are only short, they are not able to be clicked into the breadboard properly meaning that I had to just hold the pins together which means that they may not be forming a complete connection. To fix this I soldered the male headers onto the board so that the keypad could be clipped into it directly eradicating the chance of them disconnecting.

This still didn’t seem to work. The wires from the ultrasonic sensor were also wired into 2 of the pins used for the keypad inputs. Although the sensor itself was not connected I decided to cut the wires and try to upload the code again to see if that helped. When it was uploaded the code now worked again, the correct number was displayed on the serial monitor when pressed. Going forward the ultrasonic distance sensor is not needed therefore I decided to leave it disconnected.

The code was developed from just displaying the keypad number to transmitting to the Arduino the values of x, y and z which are set to different values depending on the number pressed. The options are any number, \* or # will change the motor speeds and steering angles. This was implemented by using a set if ‘else if’ statements within the ‘if key’ loop. This meant that the decision and transmission only occurred when a button was pressed instead of continuously. Originally code wouldn’t work as the quotation marks were omitted from the if statements, as the ‘key’ value is a char not an int it has to be evaluated as text instead of a number.

Graphical user interface, text, application

Description automatically generated

The screenshot shows the serial monitors of both the ESP32 and the Arduino so that we could monitor what values were being sent and received. As different buttons were pressed you can see the Arduino receiving the correctly changing values. In reality the motors were changing speed, but the servo was not moving. Upon inspection the servo seemed to have been tightened too tight meaning that it was not powerful enough to move the arm independently. The servo was taken apart and reassembled using new washers as the previous one had worn down and kept a bit looser which meant it worked properly now.

LCD display

Using the pin diagram on the data sheet I breadboarded the display with the ESP32.

Table

Description automatically generated

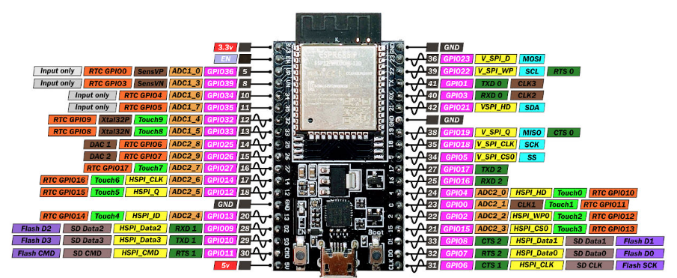


Figure 16: Pinout diagram of ESP32 microcontroller [12]

<https://www.circuitschools.com/interfacing-16x2-lcd-module-with-esp32-with-and-without-i2c/>

Used to look at which pins need to go where as well.

Table

Description automatically generated

The liquid crystal library was downloaded on Arduino to allow control of the LCD screens.

Graphical user interface, text, application

Description automatically generated

Uploaded example code ‘hello world’ from LCD library.

Graphical user interface, text, application

Description automatically generated

Wiring didn’t work so ammended which still didn’t help. Used code gotten from above link to display ‘HELLO WORLD’ with a timer as expected. Altering the message in lcd.print will change the message displayed.