International Institute of Information Technology

Introduction to IoT

Lab 4 **Spring 2022**

Overview:

In this lab session, you will learn how to use the inbuilt ADC on the esp32 to read analog inputs from

a) a water level sensor to estimate the water level in a container and

b) the capacitive touch GPIO pins to detect human touch.

Then you will implement a system that can remember the current water level on touch input and indicate if the water level goes higher or lower using a buzzer.

Water Level Sensor:

The sensor allows us to estimate the water by acting as a variable resistor whose resistance varies according to the water level.

The sensor has a series of ten exposed copper traces, five of which are power traces and five are sense traces. These traces are interlaced so that there is one sense trace between every two power traces. Usually, these traces are not connected but are bridged by water when submerged. The change in resistance corresponds to the distance from the top of the sensor to the surface of the water.

- As the sensor is immersed more, it results in better conductivity and will result in a lower resistance.
- If the sensor is immersed less, it results in poor conductivity and will result in a higher resistance.

The sensor produces an output voltage according to the resistance, by measuring which we can determine the water level.

Specifications:

- Working Voltage : DC 3-5V
- Working Current : <20mA
- Sensor Type : Analog
- Detection Area : 40 mm x 16 mm
- Size : 65 mm x 20 mm x 8 mm
- Humidity: 10% -90% non-condensing

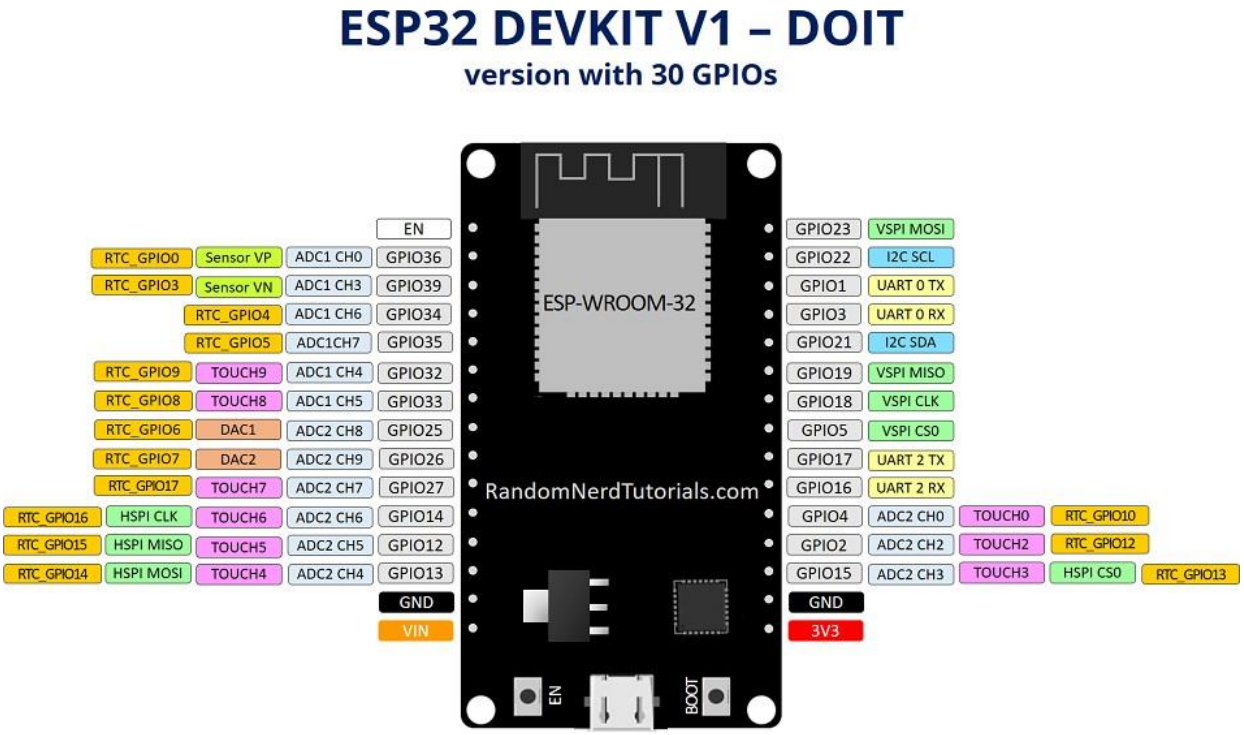
Pinout
S (Signal) pin is an analog output that should be connected to one of the analog inputs on your ESP32 (see board pinout).

+ (VCC) pin supplies power for the sensor. It is recommended to power the sensor with between 3.3V – 5V. Please note that the analog output will vary depending on what voltage is provided for the sensor. However, having power applied to the probe constantly speeds the rate of corrosion significantly. To overcome this, we recommend that you do not power the sensor constantly, but power it only when you take the readings. An easy way to accomplish this is to connect the VCC pin to a digital output pin and set it to HIGH or LOW as per your requirement.

– (GND) is a ground connection.

Capacitive Touch Input

The ESP32 has 10 capacitive touch GPIOs. These GPIOs can sense variations in anything that holds an electrical charge, like the human skin. So they can detect variations induced when touching the GPIOs with a finger. These pins can be easily integrated into capacitive pads, and replace mechanical buttons. Additionally, the touch pins can also be used as a wake-up source when the ESP32 is in deep sleep. These pins have been labelled as TOUCH in violet below.



Experiment Part 1: Read the value from the water level sensor and output to the serial monitor.

Hardware Required:

- Breadboard
- ESP32
- Water level Sensor
- Jumper wires

Circuit Connections:

1. Connect - pin on the sensor to gnd on the esp32
2. Connect + pin on the sensor to any GPIO pin
3. Connect S pin on the sensor to any GPIO pin which supports ADC (see board pinout).

PseudoCode:

1. Define sensor power pin and signal pin
2. In the Setup function:
    a. Use Serial.begin() to begin the serial communication
    b. Declare sensor power pin as digital output pin and set it to LOW.
3. In the Loop block:
    a. Power on the water level sensor
    b. Read the sensor value using analogRead(sensorpin_no)
    c. Print the value on the serial monitor

Immerse the sensor in a small vessel containing water and see how the reading corresponds to water level.

Experiment Part 2:
Read the value from a capacitive touch pin and print it to the serial monitor. Activate a buzzer when a touch is detected.

Additional Hardware Required:

- Passive Buzzer MH-FMD
- Jumper wire

Circuit Connections:

1. Connect the shorter pin on the buzzer to GND on the ESP32 and the other pin to a GPIO pin on the ESP32
2. Connect a wire to one of the capacitive touch pins on the ESP32 and leave the other end free to touch.

PseudoCode:

1. Define the buzzer and touch pins
2. In the Setup function:
    a. Use Serial.begin() to begin the serial communication
    b. Declare buzzer pin as digital output pin and set it to LOW.
3. In the Loop function:
    a. Read the value from the touch pin using touchRead() and print to serial monitor.
    b. If the value is less than a certain threshold (calibrate to detect touch consistently) sound the buzzer using tone() function.

Experiment Part 3:
Combine the previous two experiments to create a system that recordsthe current water level value when the pin is touched and sounds the buzzer if the water level deviates beyond a threshold.

PseudoCode:

1. Define sensor power pin, signal pin, buzzer pin and thresholds
2. Declare a variable to store required_water_level_value
3. In the Setup function:
    a. Use Serial.begin() to begin the serial communication
    b. Declare sensor power pin as digital output pin and set it to LOW.
    c. Declare buzzer pin as digital output pin and set it to LOW.
4. In the Loop block:
    a. Power the water level sensor
    b. Read the water level sensor value using analogRead()
    c. Power off the water level sensor
    d. Read touch pin value
    e. If touch is detected, update the value of required_water_level

f. If the water level is above or below the required_water_level beyond the threshold print a suitable message and sound the buzzer.
g. Apply appropriate delay

********** THE END **********