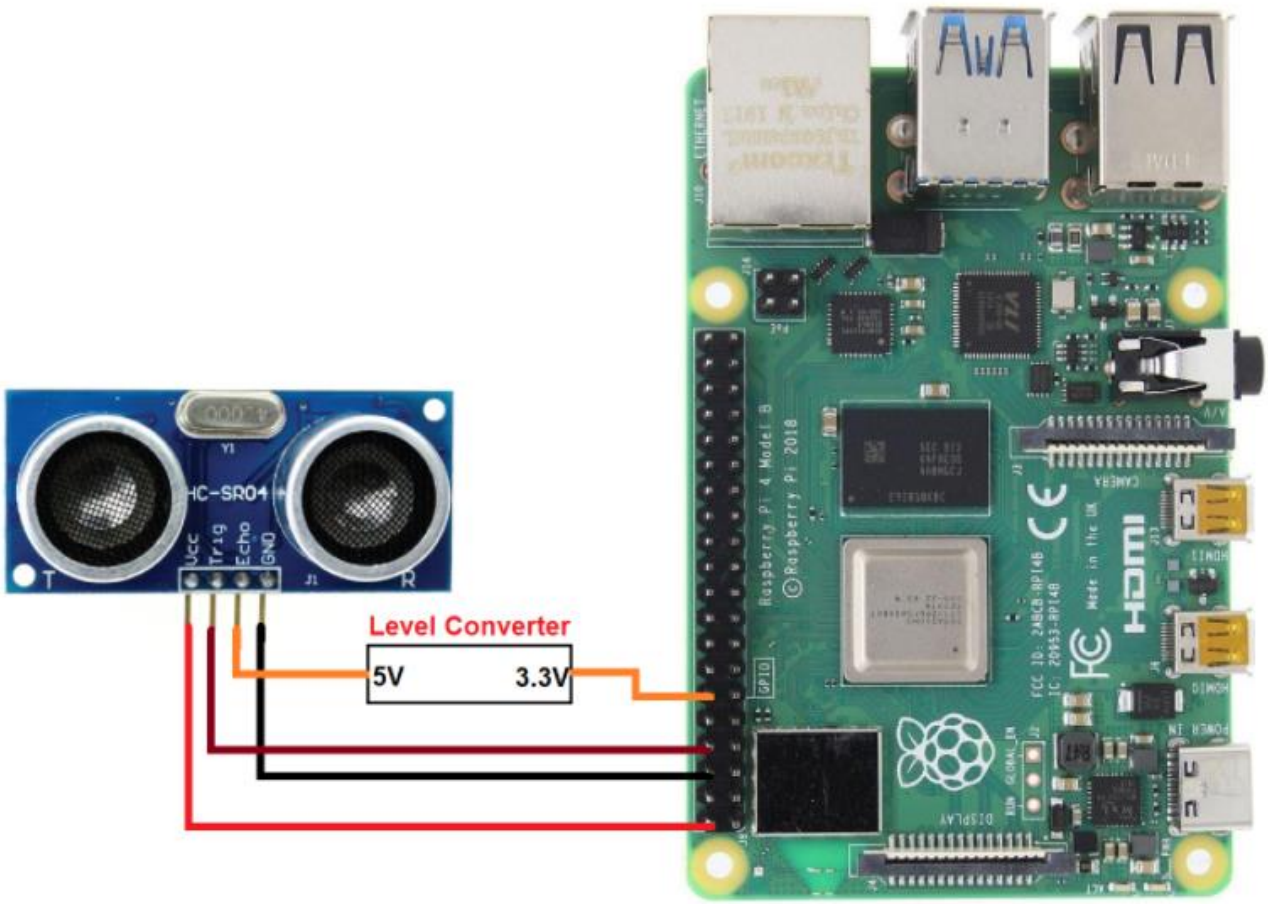
Smart Parking System

PHASE 3: Development Part 1

In this part we will start building the IoT sensor system and Raspberry Pi integration.
Configure IoT sensors to detect parking space occupancy.
Write Python scripts on Raspberry Pi to collect data from sensors and send it to the cloud or mobile app server.

Interfacing of ultrasnic sensors with Raspberry Pi

Ultrasonic sensors are used for distance measurement purposes. Typically, we use ultrasonic sensors in obstacle prevention robots and DIY radar projects. Similar sensors like IR sensor, LIDAR, sonar sensor are also available in the market which can be used for similar purposes. Ultrasonic sensor requires 5 volts, we will need to connect resistors to interface it with the 3.3V GPIO pins.



CODE for interfacing of ultrasnic sensors with Raspberry Pi

The code starts from importing modules that can be used to work with the raspberry pi GPIO pins.

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO_TRIG = 11
GPIO_ECHO = 18
GPIO.setup(GPIO_TRIG, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)
GPIO.output(GPIO_TRIG, GPIO.LOW)
Time.sleep(2)
GPIO.output(GPIO_TRIG, GPIO.HIGH)
Time.sleep(0.00001)
GPIO.output(GPIO_TRIG, GPIO.LOW)
while GPIO.input(GPIO_ECHO)==0:
start_time = time.time()
while GPIO.input(GPIO_ECHO)==1:
Bounce_back_time = time.time()
pulse_duration = Bounce_back_time - start_time
distance = round(pulse_duration * 17150, 2)
print (f"Distance: {distance} cm")
GPIO.cleanup()
```

Detecting Parking Space Availability

We will be publishing our parking space's availability in the main function of our python script. This will be the function that is executed once the script is started. The main function must do the following actions in order.

1.Set up the sensor
2.Perform an initial check on the availability of the parking space
3.Check to see if status has changed (every 5 seconds): If it has changed then publish message of new status. Store this status as the new last updated status.

To set up the sensor we will call the function we defined earlier, setup_sensor(). Next it will do an initial check to see if the parking space is available or vacant. We will define method initial_check() to do this.

We will then have a loop that is executed every 5 seconds to check the new distance reading of the car. If the distance reading obtained by function get_distance() is greater than the length of the parking space, then we can infer that the parking space is vacant. If it is less than the length of the parking space, then we know that the parking space is being occupied by a car. Here 7 is the length of the parking lot.

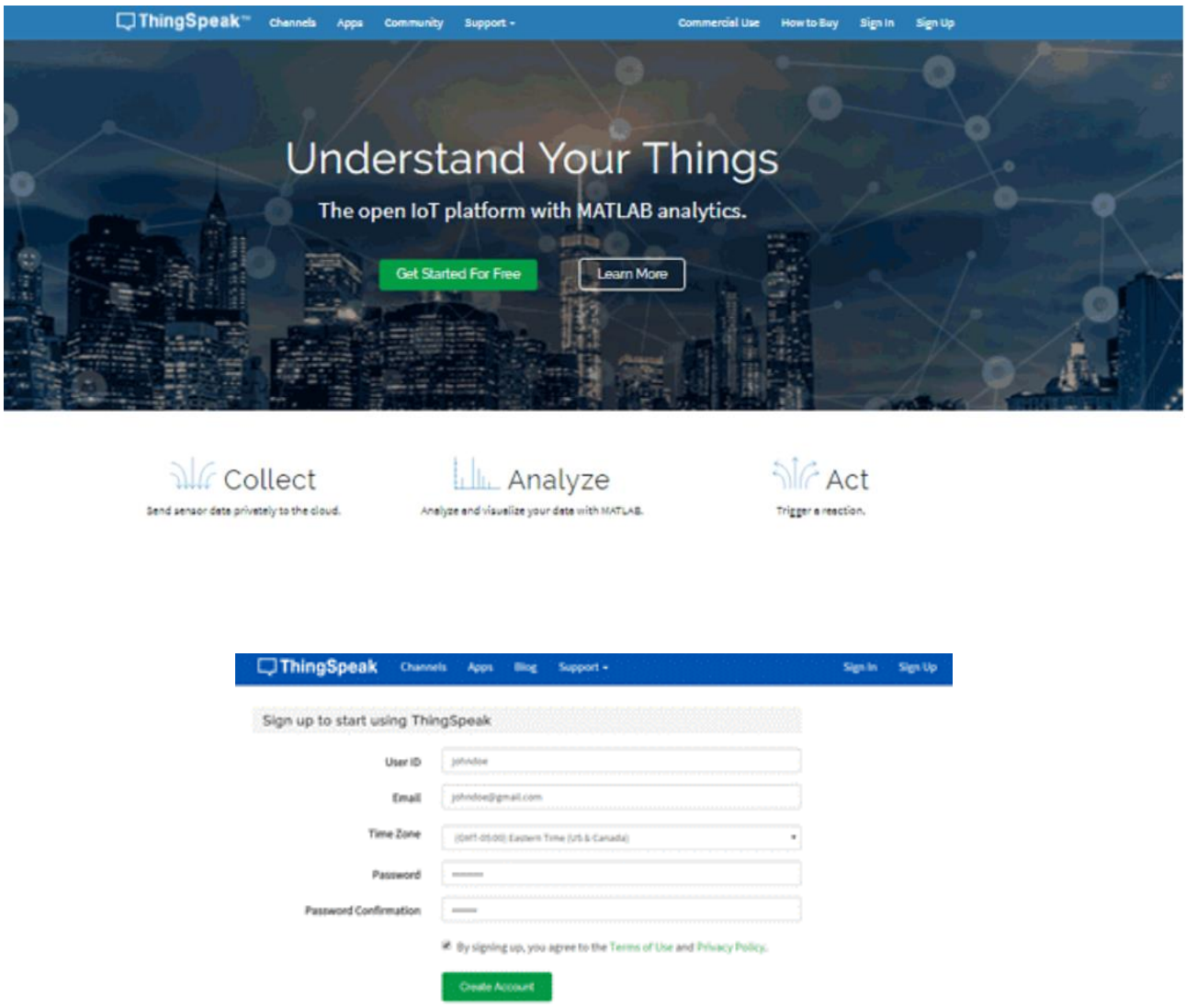
Code for updating the status of the parking space:

```
if __name__ == '__main__':
  setup_sensor()
  initial_check()
  while True:
    if (occupied and (get_distance() >= 7)) or (not occupied and (get_distance() < 7)):
        // TODO toggle the availability of the parking space and publish new status
    time.sleep(5)

def initial_check():
  occupied = True if get_distance() < 7 else False
  subprocess.Popen(["mosquitto_pub", "-h", "beam.soracom.io", "-p", "1883", "-t",
"parking_spot", "-m", convertToJsonString(occupied)], stdout=subprocess.PIPE)
  print(occupied)

def convertToJsonString(occupied):
  dictionary_object = {
    "occupied": occupied
  }
  return json.dumps(dictionary_object)

occupied = not occupied
subprocess.Popen(["mosquitto_pub", "-h", "beam.soracom.io", "-p", "1883", "-t",
"parking_spot", "-m", convertToJsonString(occupied)], stdout=subprocess.PIPE)
print(occupied)

def close(signal, frame):
  print("
Turning off ultrasonic distance detection...
")
  GPIO.cleanup()
  sys.exit(0)
signal.signal(signal.SIGINT, close)
```

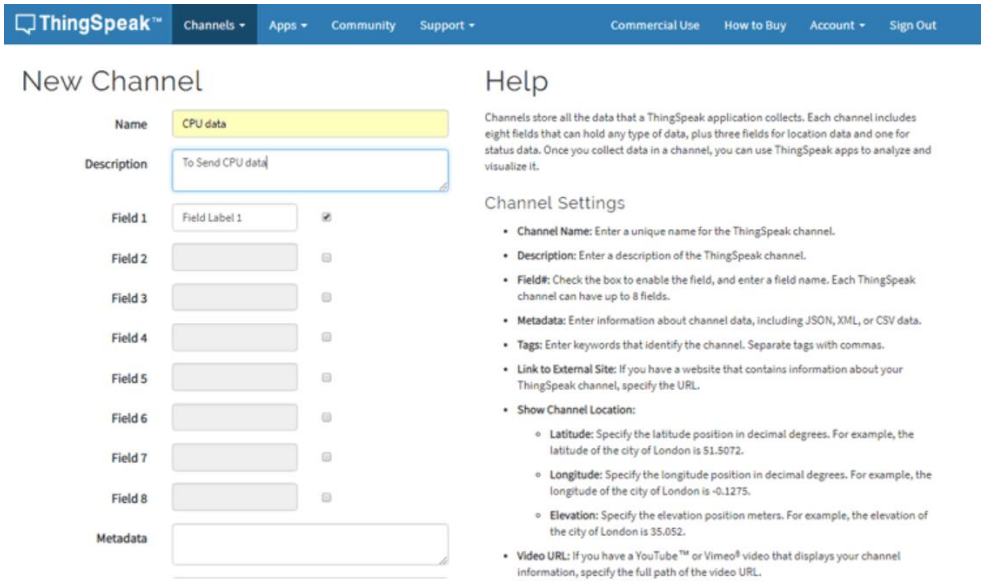
Steps for uploading Raspberry Pi Data on Cloud

Step 1: Signup for ThingSpeak

For creating channel on ThingSpeak, sign up on ThingSpeak. In case we already have account on ThingSpeak, we sign in using your id and password, if not we create an account

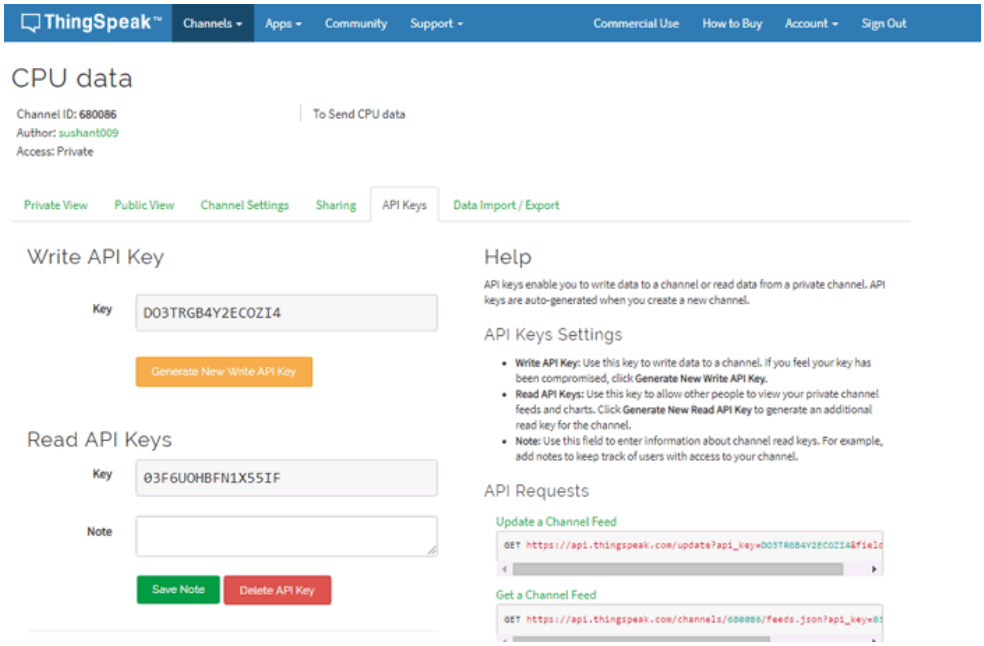


After creating an account, we create the channel



Step 3: Getting API Key in ThingSpeak

To send data to ThingSpeak, we need an unique API key, which we will use later in our python code to upload our data to ThingSpeak Website.



CODE for uploading the data collected from the sensor to Thingspeak

```
from masterclass import *
distance = 11

w_key = 'AZRMN9ZP5FKLXYLR'
r_key = '17FSCQ4FTX6V4VWM'
channel_id = 83234

ob = Thingspeak(write_api_key=w_key, read_api_key=r_key,
channel_id=channel_id)
ob.post_cloud(value1= distance)
```

Thus the development part 1 for the smart parking system using IoT is done and documented successfully.