Executing UAV/drone control on Raspberry Pi

Ashita 2019028
Rishi 2017260

Links

https://github.com/ashcode028/UAV-RPI-Control/

https://docs.google.com/document/d/18BHOcjUK4YEGwr3YdviBCbj2okhxxjBthUeENKONWTQ/edit

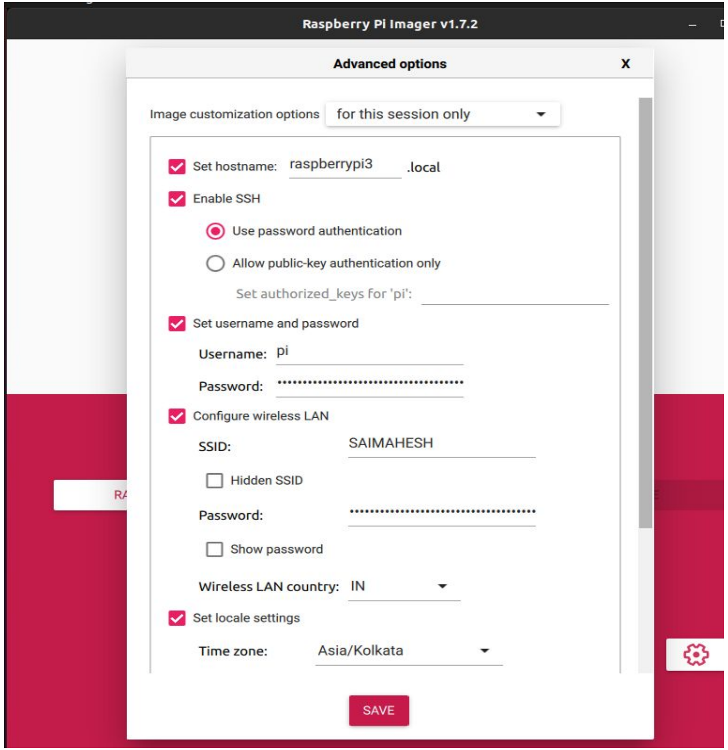
Motivation

With the emerging 5G/6G technologies, drones or unmanned aerial vehicles (UAVs) are widely expected to result in enabling new types of applications and services in cities. Such applications include delivery of items both at the intermediate points in the supply-chain as well as the last mile, sensing of various environmental parameters such as air pollution, mosquito breeding sites as well as infrastructural usage like the conditions of roads and traffic loads. They can also provide connectivity services to users by positioning themselves in proper locations so that they can serve as WiFi hotspots.

Background

These applications typically use a network of drones that are controlled either by a centralized or local controller. Controlling such a network of drones, however, brings in a number of challenges. First, most of these applications require coordination among the drones. A network of drones use the cellular network to send the sensed data, as well as receive the control information. However, for such a system to work reliably, it is essential for the drones to get the control information within a specific end-to-end latency

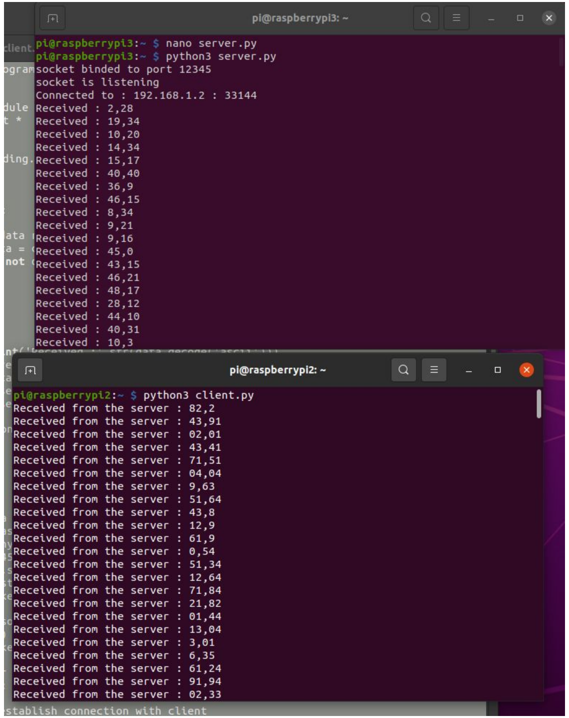
Setting up Rpi:



- Install RPI imager, *sudo snap install rpi-imager*. Apt-get package is broken , don’t use it.
- Once it’s downloaded , select os and the storage devices to be written.
- There is an settings icon(advanced options) appearing once u select the above two, once we click on it, we can configure ssh, hostname , password authentication.
  - We can also manually configure wpa_supplicant without using the rpi-imager as well but this method is quite time saving for first time.
- If os is already installed but only ssh enable is left, then follow the steps below,
  - after inserting the micro sd card there is a folder named “boot”
  - Run following commands:
    - touch ssh
    - touch wpa_supplicant.conf, its contents would be

```
country=IN # replace with your country code
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
network={
   ssid="WIFI_NETWORK_NAME"
   psk="WIFI_PASSWORD"
   key_mgmt=WPA-PSK
}
```

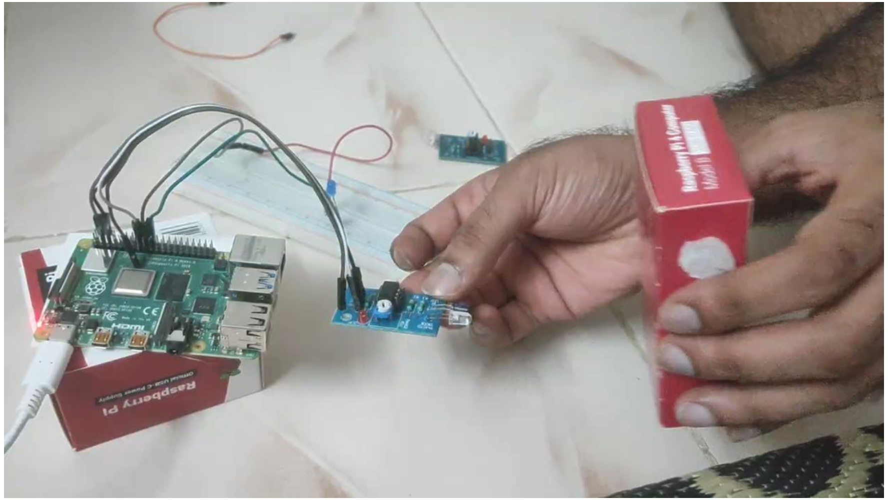
Socket program Laptop as access point

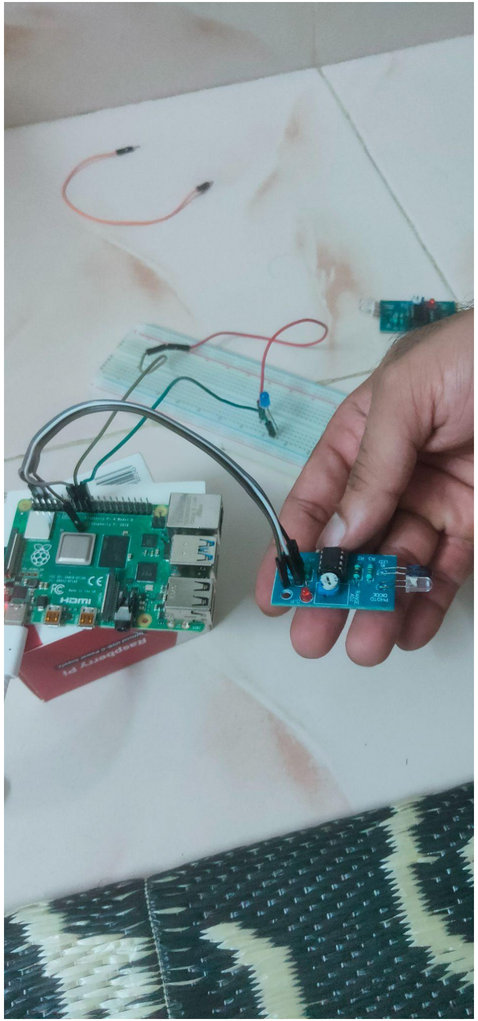


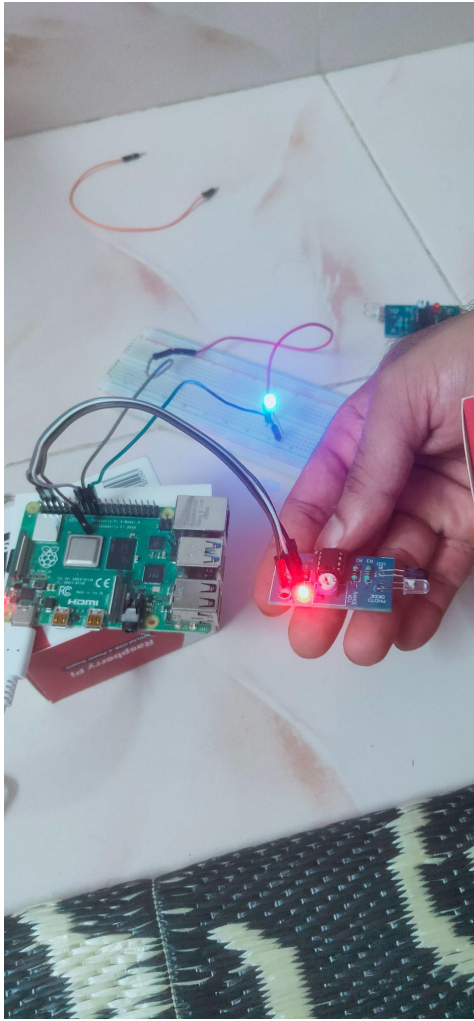
- ➔ Sending dummy temperature and humidity values from one rpi to another.
- ➔ Can save some data into file and send that file using iperf across

Proximity setup

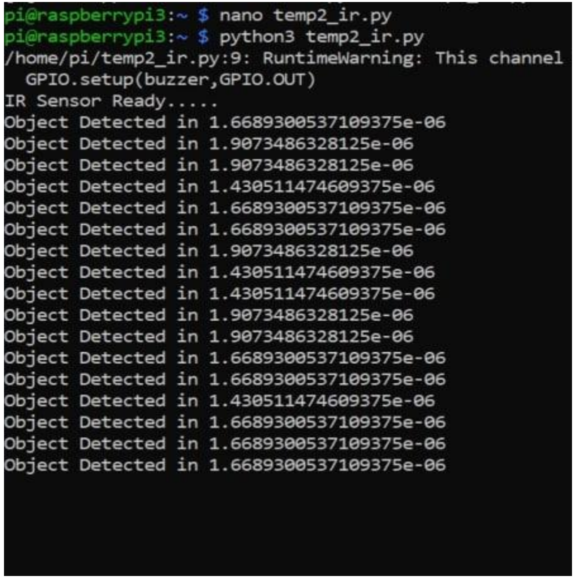
- ➔ LCD (or can also be buzzer) used to send a signal to detect object.
- ➔ Working smoothly







Proximity sensor working



```
pi@raspberrypi3:~ $ nano temp2_ir.py
pi@raspberrypi3:~ $ python3 temp2_ir.py
/home/pi/temp2_ir.py:9: RuntimeWarning: This channel
  GPIO.setup(buzzer,GPIO.OUT)
IR Sensor Ready.....
Object Detected in 1.6689300537109375e-06
Object Detected in 1.9073486328125e-06
Object Detected in 1.9073486328125e-06
Object Detected in 1.430511474609375e-06
Object Detected in 1.6689300537109375e-06
Object Detected in 1.6689300537109375e-06
Object Detected in 1.9073486328125e-06
Object Detected in 1.430511474609375e-06
Object Detected in 1.430511474609375e-06
Object Detected in 1.9073486328125e-06
Object Detected in 1.9073486328125e-06
Object Detected in 1.6689300537109375e-06
Object Detected in 1.6689300537109375e-06
Object Detected in 1.430511474609375e-06
Object Detected in 1.6689300537109375e-06
Object Detected in 1.6689300537109375e-06
Object Detected in 1.6689300537109375e-06
```

- ➔ Will be used for obstacle detection
- ➔ Requires instant trigger , rather than like pub/sub model
- ➔ If the object detection program runs for a specific time , we can send timestamps of the obstacle detected.
- ➔ Mean of the time taken for one object detection ~**1.4microseconds** : ~0.7Million detections/ sec
- ➔ Mean of time taken to object detection transmission:**~5.1microseconds** :~ 0.2Million detections/sec

File names:

Temp_ir.py: Object detection using proximity sensor/IR

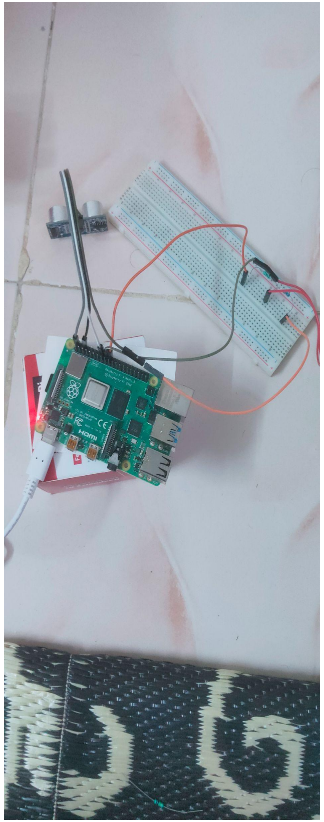
Temp_ultra.py : distance measurement using ultrasonic sensor

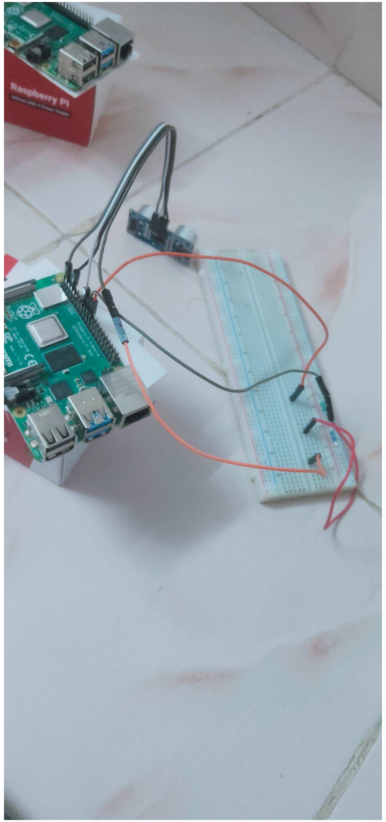
Client.py : client sending dummy data to server using sockets

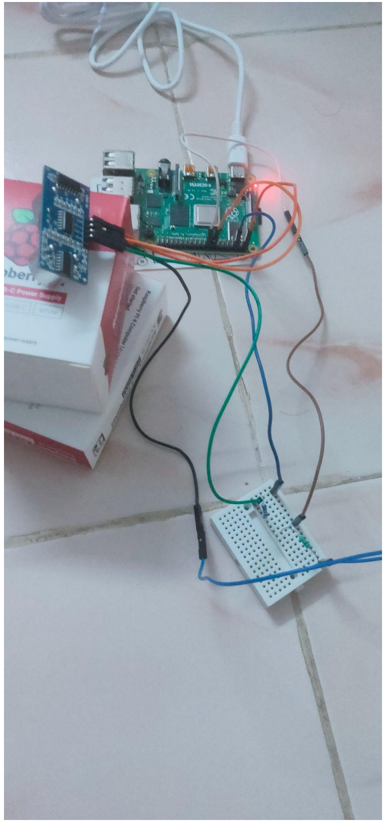
Server.py : server receiving dummy data from client using sockets

Ultrasonic setup

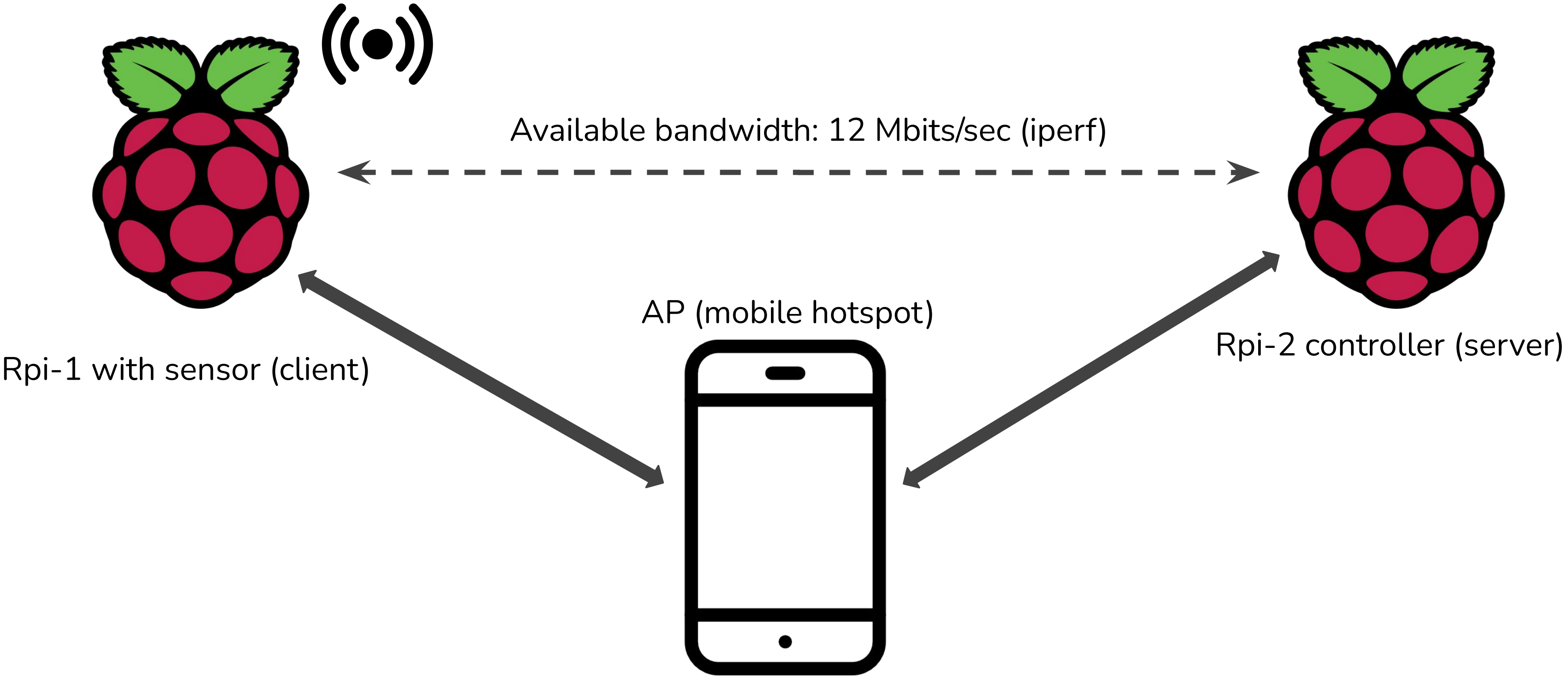
- ➔ Resistors required : 10k ohms , 4.5k ohm to achieve desired voltage.
- ➔ Problems faced : pin setup, echo not receiving.







Setup-1



Available bandwidth: 12 Mbits/sec (iperf)
AP (mobile hotspot)
Rpi-1 with sensor (client)
Rpi-2 controller (server)

Data Format

1. Each client regularly use ultrasonic sensor to measure the distance.
2. This distance measurement is paired with timestamp and then sent to the server.
    a. Timestamp ← Number of seconds passed since epoch (Unix timestamp) : type float

Packet Format (16 Bytes)

Distance (in cm)	Unix Timestamp (s)
Double precision float (8 bytes)	Double precision float (8 bytes)

Strategies

- Single threaded tcp server and client
- Single threaded udp server and client
- Basic message queue based multi threaded tcp client.

Implementations

simple_tcp

```python
import socket
from ultra import UltraApp
import struct
import sys
import time

if __name__ == "__main__":
    app = UltraApp(18, 24)
    total = 10000
    try:
        sock = socket.create_connection((sys.argv[1], sys.argv[2]))
        app.setup()
        ctr = 0

        while ctr < total:
            dist = app.distance()
            tsp = time.time()

            to_send = struct.pack("!dd", dist, tsp)
            sock.sendall(to_send, len(to_send))
            ctr += 1
    finally:
        print("[+] Closing App")
        app.cleanup()
        sock.close()
```

simple_udp

```python
import socket
from ultra import UltraApp
import struct
import sys
import time

if __name__ == "__main__":
    app = UltraApp(18, 24)
    total = 10000
    addr = (sys.argv[1], int(sys.argv[2]))
    end_msg = struct.pack("!dd", float("nan"), float("nan"))
    try:
        sock = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
        app.setup()
        ctr = 0

        while ctr < total:
            dist = app.distance()
            tsp = time.time()

            to_send = struct.pack("!dd", dist, tsp)
            sock.sendto(to_send, addr)
            print(ctr)
            ctr += 1

        sock.sendto(end_msg, addr)
    finally:
        print("[+] Closing App")
        app.cleanup()
```

Implementations

bmq_tcp

```python
import socket
from ultra import UltraApp
import struct
import sys
import time
import queue
import threading

q = queue.Queue()


def tcp_worker(addr):
    try:
        sock = socket.create_connection(addr)
        print("[+] Connected")

        while True:
            item = q.get()
            sock.sendall(item, len(item))
            q.task_done()
    finally:
        sock.close()


def app_worker():
    app = UltraApp(18, 24)
    total = 10000
    try:
        app.setup()
        ctr = 0
        while ctr < total:
            dist = app.distance()
            tsp = time.time()
            to_send = struct.pack("!dd", dist, tsp)
            q.put(to_send)
            ctr += 1
    finally:
        app.cleanup()

if __name__ == "__main__":
    addr = (sys.argv[1], int(sys.argv[2]))
    app_thrd = threading.Thread(target=app_worker).start()
    worker_thrd = threading.Thread(target = tcp_worker, kwargs={"addr": addr}, daemon=True).start()

    q.join()
```

Throughput calculation

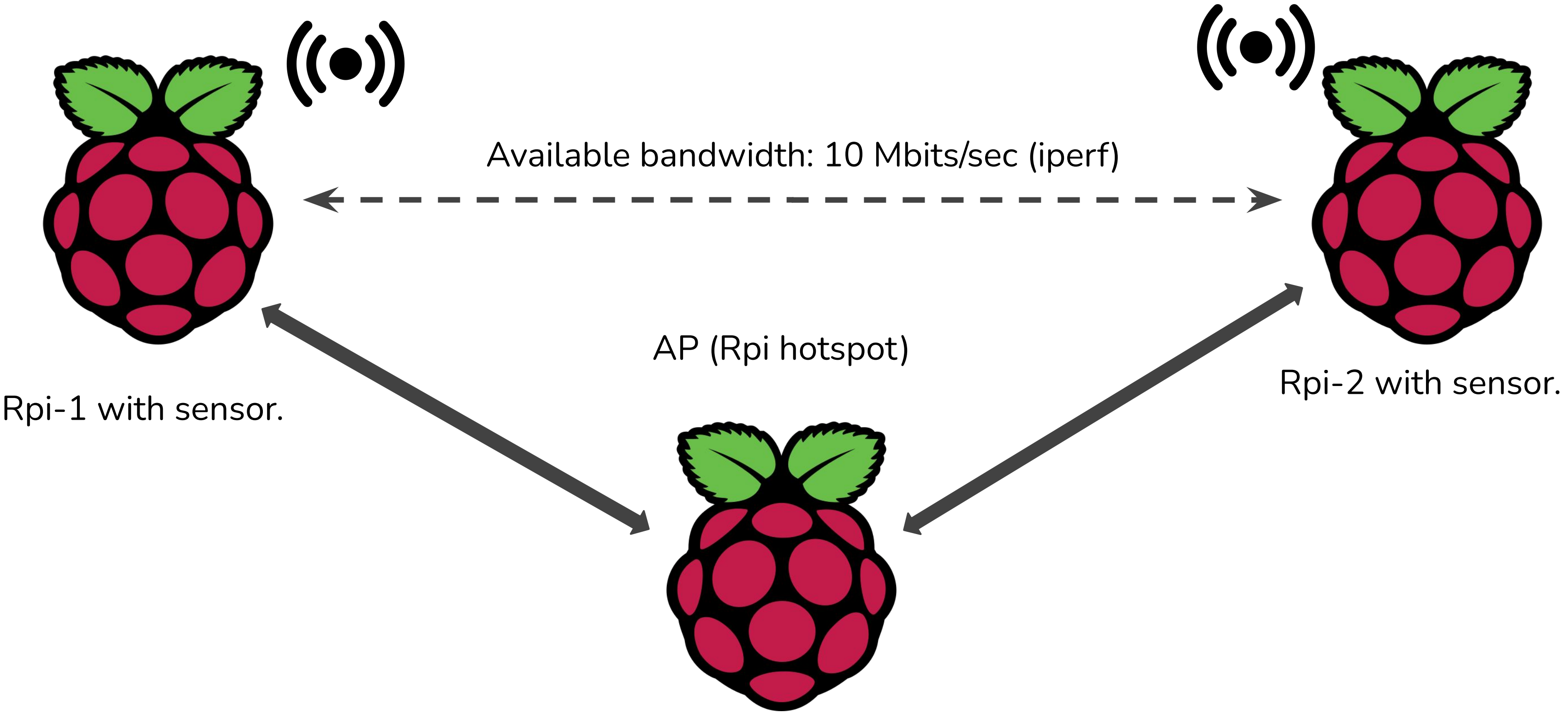
```python
def calc_throughput(tsps):
    if len(tsps) == 0:
        return float('nan')
    delta = 0.1
    base = tsps[0]
    datapoint = 1
    tpts = []
    for i in range(1, len(tsps)):
        curr = tsps[i]
        if (curr - base >= (1 - delta)):
            tpts.append(datapoint)
            base = curr
            datapoint = 1
        else:
            datapoint += 1

    return sum(tpts)/len(tpts)
```

Results

Strategy	Datapoints/sec	Throughput (Kb/sec)
Single Thread tcp	366.55	5.864
Single Thread udp	325.105	5.249
Basic Message Queue	374.153	5.986

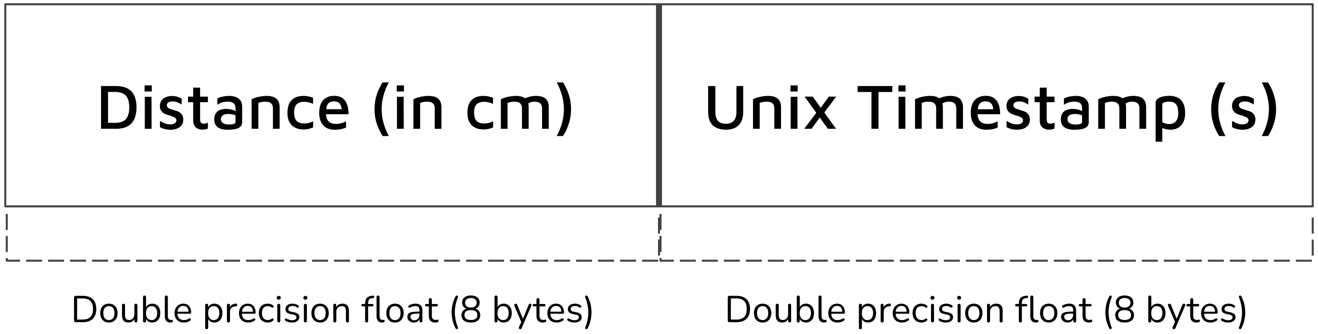
Setup-2 (Both client as well as server)



Data Format

1. Each client regularly use ultrasonic sensor to measure the distance.
2. This distance measurement is paired with timestamp and then sent to the server.
   a. Timestamp ← Number of seconds passed since epoch (Unix timestamp) : type float

Packet Format (16 Bytes)



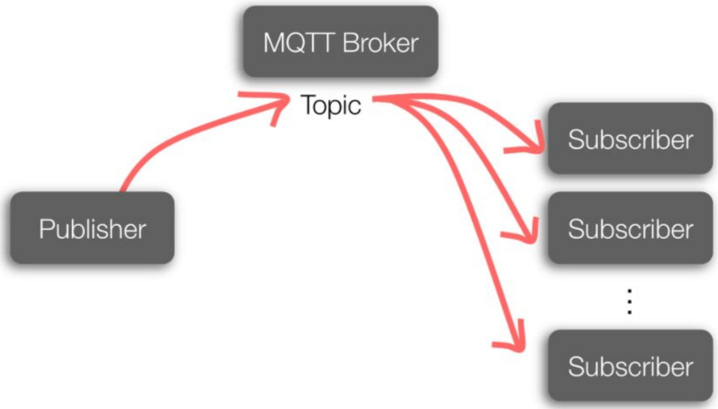
Results

Strategy	Datapoints/sec	Throughput (Kb/sec)
Single Thread tcp	355.821	5.693
Single Thread udp	325.105	5.249
Basic Message Queue	365.363	5.840

Conclusions

- ➔ Network optimizations would make more sense probably for high cost, higher throughput sensors.
- ➔ This type of system could easily scale as just a small amount of bandwidth is required to stream sensor data.
- ➔ If it was camera sensor : to send gray-scale image it would ~512Kbps for one image. So similarly if we need video i.e ~40 images(per frame) then ~40Mbps for one video.
- ➔ We can use some compression techniques to compress those images after capturing, reduce the throughput for a single video transfer

Conclusions & Future Work



- ➔ Decide over type of connection(Tcp or udp), model of sending and receiving to controller(pub/sub model or instantly).
- ➔ An application running on Rpi which can implement publisher/subscriber model. (Eg MQTT)

References/Findings

- Rpi os setup:

https://www.youtube.com/watch?v=1-IiLA8chCA
https://www.youtube.com/watch?v=63yw7b0NuWc
https://roboticsbackend.com/enable-ssh-on-raspberry-pi-raspbian/
https://www.seeedstudio.com/blog/2021/01/25/three-methods-to-configure-raspberry-pi-wifi/
https://eng.ox.ac.uk/computing/projects/programmable-hardware/p4pi/

- Sensors rpi setup:

https://robu.in/raspberry-pi-ultrasonic-sensor-interface-tutorial/
https://www.electronicshub.org/raspberry-pi-ultrasonic-sensor-interface-tutorial/

- Sending data across rpi:

https://www.youtube.com/watch?v=QihjI84Z2tQ
https://soumilshah1995.blogspot.com/2019/04/server-and-client-send-actual-sensor.html
https://forums.raspberrypi.com/viewtopic.php?t=111220

- Iperf: based on type of connection and data to be sent

https://netbeez.net/blog/raspberry-pi-and-distributed-network-monitoring-iperf/
https://www.dell.com/support/kbdoc/en-in/000139427/how-to-test-available-network-bandwidth-using-iperf

- Test scripts and tools for latency, bandwidth , throughput

https://www.binarytides.com/linux-commands-monitor-network/
https://chromium.googlesource.com/chromiumos/platform/factory/+/HEAD/py/test/pytests/wifi_throughput.py

- Pub/sub setup

https://www.instructables.com/Installing-MQTT-BrokerMosquitto-on-Raspberry-Pi/