COMMUNICATIONS

Wang Minxi John (2101925)
Ng Zi Hao (2102893)
Low Li Pin (2101542)

Table of Contents

01

Overview

Overview

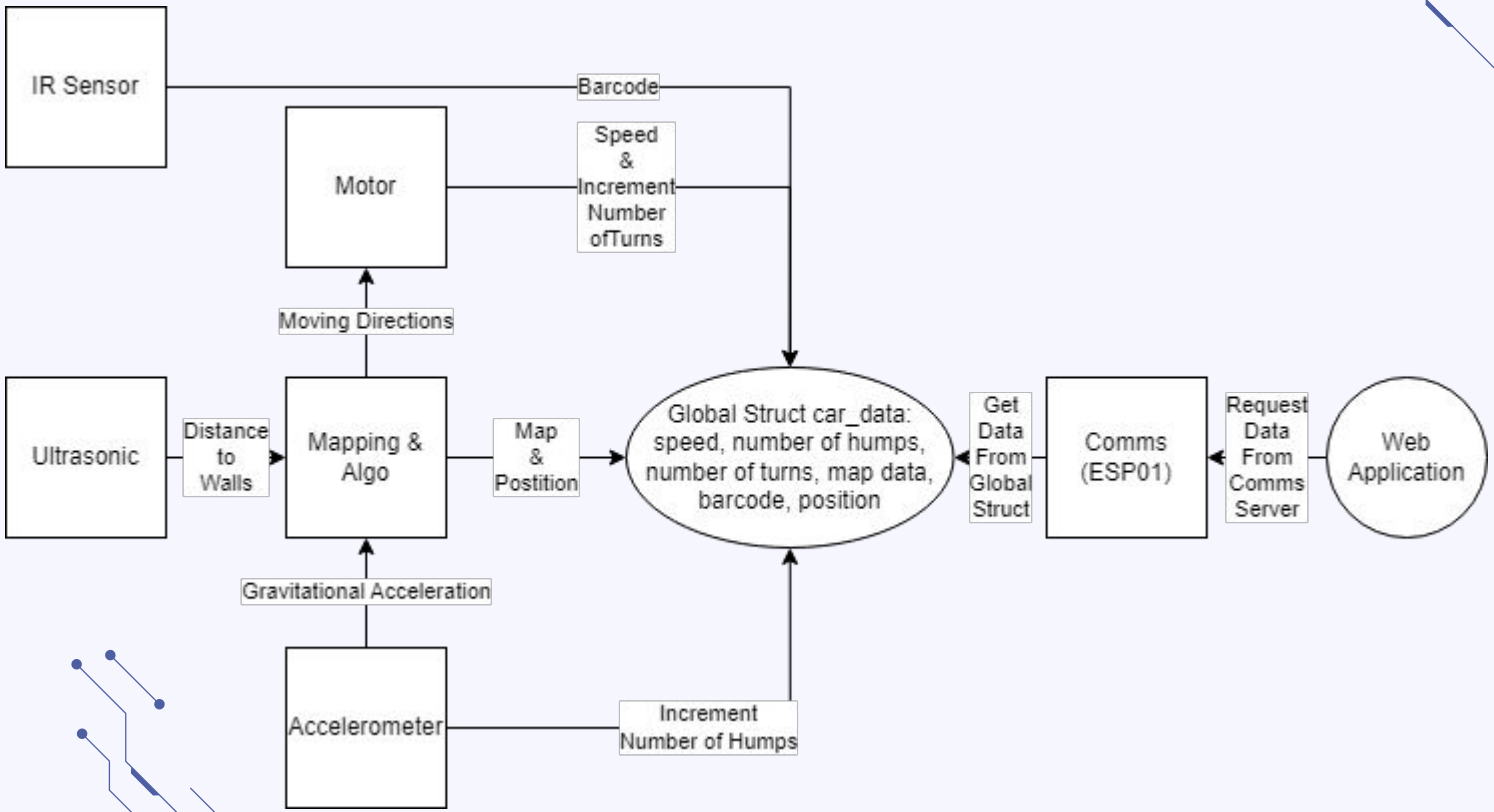
- Pico Board

- Communication protocol used:
  - PICO -> ESP01 (UART)
  - PICO -> HC05 (UART)
  - PICO -> M5STICKC PLUS (UART)
  - MQTT (M5Stick)

Overview

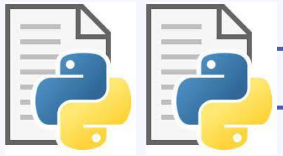
- Flask Web Application displaying data collected from the car
  - Functions:
    - Send starting node
    - Send direction of the car (left, front, right, back)
    - Display event (hump detected, turning, etc)
    - Display distance travelled
    - Display car speed
    - Display barcode information
    - Display number of turns
    - Display map nodes

Program Flowchart



02

UART
(M5StickC Plus)



Python Files

Publish

Subscribe

HIVEMQ

MQTT Broker

Publish

Subscribe

HIVEMQ
MQTT CLIENT

Publish

Subscribe

Subscribe

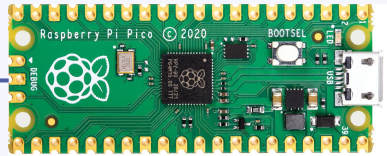
2.4 GHz (only)
WiFi Hotspot

Connected to

M5StickC Plus

Web Application

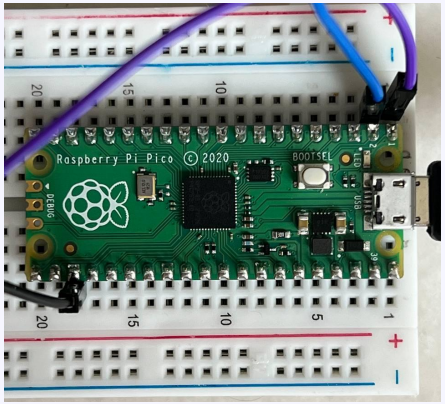
Transfer
data via TX



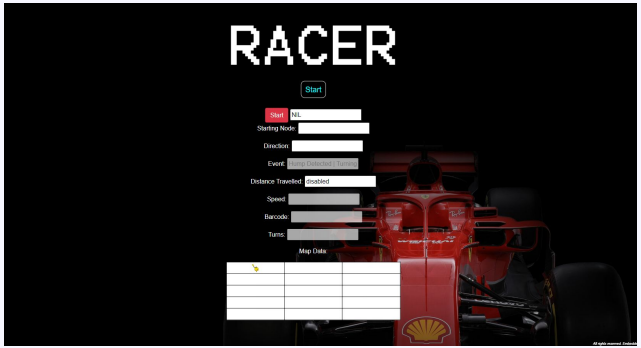
Pico Board

Service is used to send data from Pico to M5StickC Plus to Web Application for data such as speed

Setup


Raspberry Pi Pico © 2020
BOOTSEL


GND
5V
G26
G0
BAT
3V3
5V


RACER
Start
Start
NIL
Starting Node:
Direction:
Event:
Distance Traveled:
Enabled
Speed:
Barcode:
Turns:
Map Data:


MQTT is connected
Recieving messag
e: 10
Sending message
to MQTT
M5

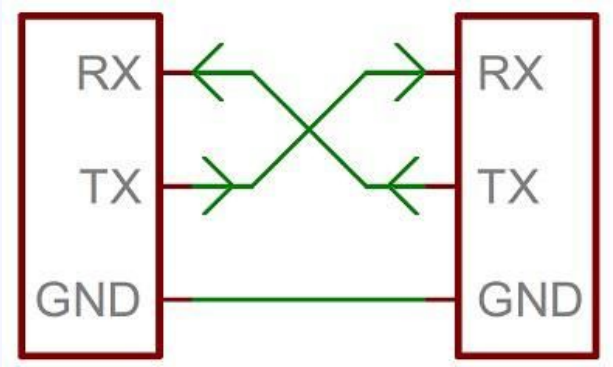
Setup

- M5StickC Plus
  - TX -> G32
  - RX -> G33
  - Databits -> 8
  - StopBits -> 1
  - ParityBits -> None

- Pico Board
  - TX -> G1
  - RX -> G0
  - Databits -> 8
  - StopBits -> 1
  - ParityBits -> None

- UART0

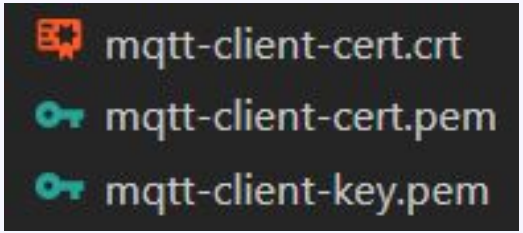
M5StickC Plus **Pico Board**



Setup

- MQTT
  - HiveMQ Broker
  - Broker URL
    - 3eb3b23922da49208766dd4507ecd30c.s1.eu.hivemq.cloud
  - Created 10 user accounts
    - user-1 -> user-10
    - SSL enabled (for secure MQTT)
    - SSL protocol (TLSv1.2)

- Web Application
  - CA_CERTS
  - CERTFILE
  - KEYFILE



Functions

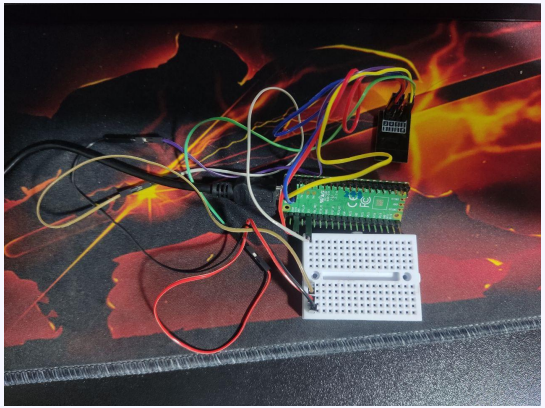
	Main Functions
M5StickC Plus	`initiate_UART()`
	`if uart1.any():`
Pico Board	`uart_puts();`
Web Application	`@mqtt_client.on_message()`

Video Demo

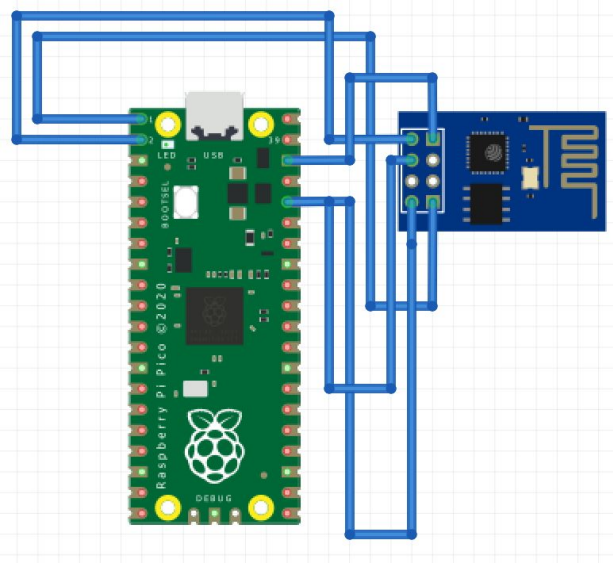
03

UART (ESP8266)

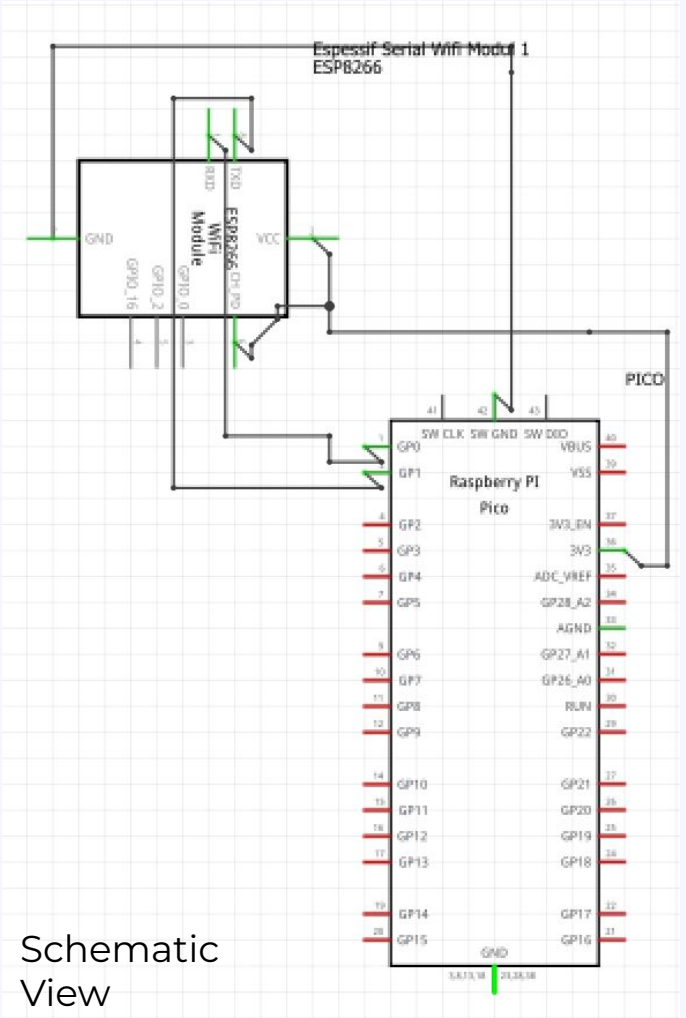
Setup



Actual Setup



Breadboard View



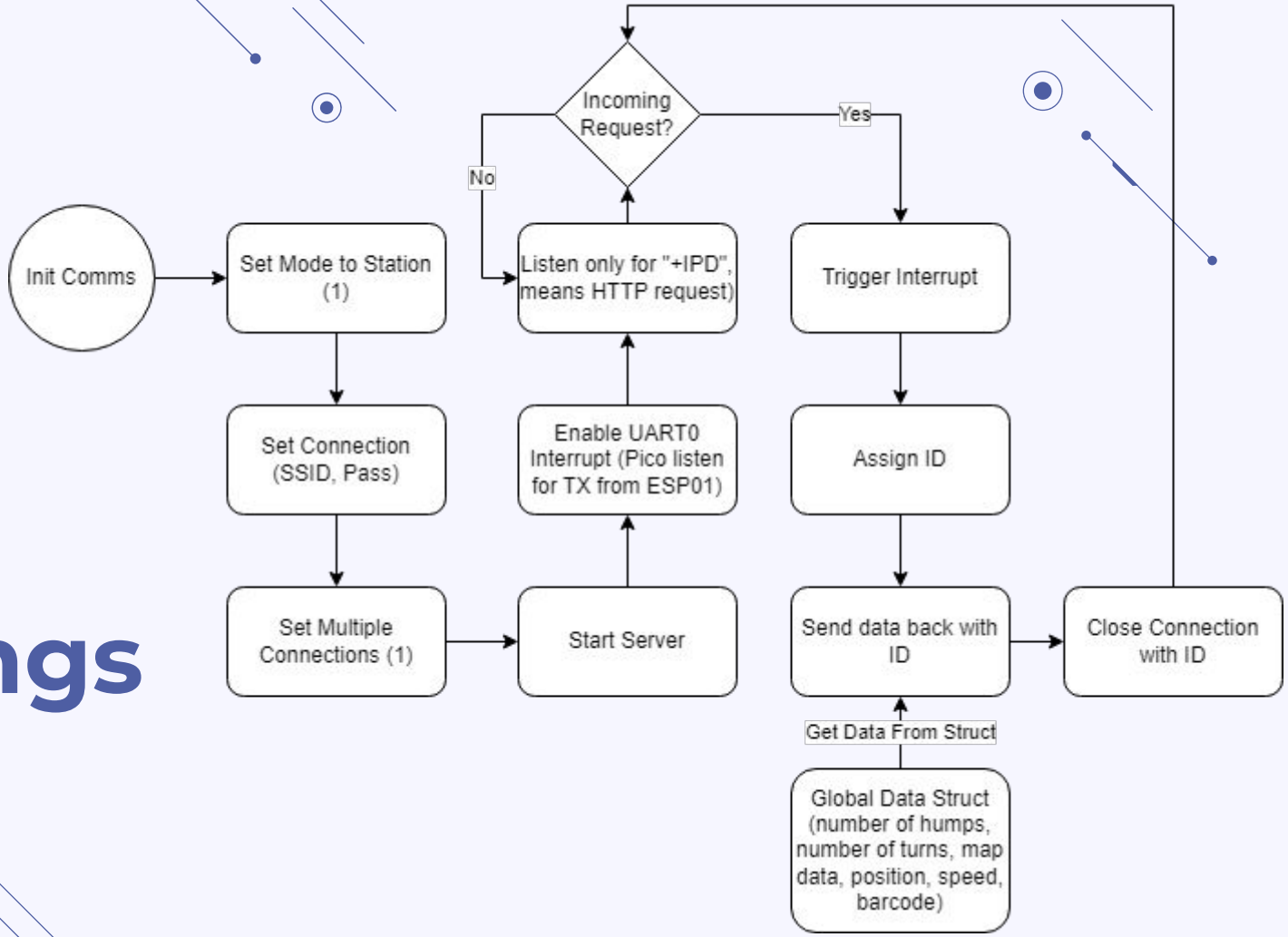
Schematic View

Config

Pico Config-
UART0
TX_PIN 0, RX_PIN 1
Baud Rate 115200
Databits 8 | Stopbits 1 |Parity None

Pin Connection-
Pico GP0 (TX) -> ESP01 (RX)
Pico GP1 (RX) -> ESP01 (TX)
Pico 3V3-> ESP01 (VCC)
Pico GND -> ESP01 (GND)
Pico 3V3 -> ESP01 (EN)

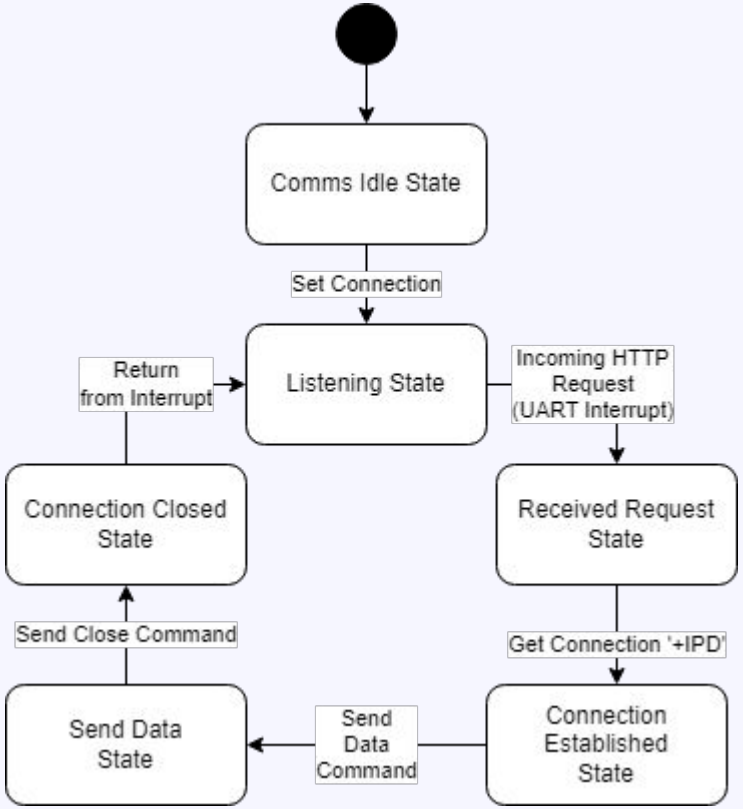
Inner Workings



Commands

1) Set up UART0 on Pico
2) Set mode with `"AT+CWMODE=1”`
3) Set connection with `"AT+CWJAP=”ssid”,”password””`
4) Get IP with `"AT+CIFSR”`
5) Get ID by listening to first occurrence of “+IPD” by using strstr to check UART0 RX interrupt string
6) Set multiple connections with `”AT+CIPMUX=1”`
7) Start server with `"AT+CIPSERVER=1,80”`
8) Send data with `”AT+CIPSEND=<ID>”, wait for “>”, then send data`
9) Close IPD connection with `“AT+CIPCLOSE=<ID>”`

Finite State Machine

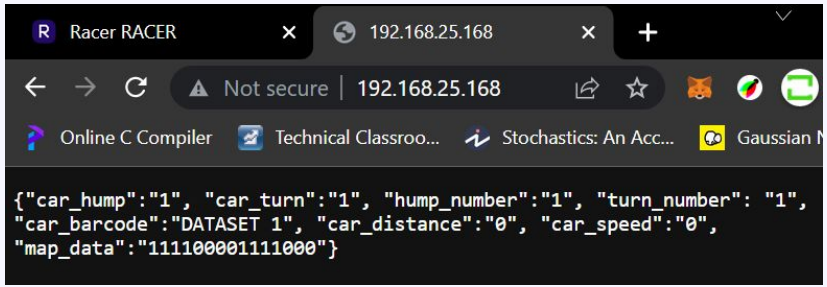


Testing

Conducted black box testing

1) Testing Criteria
    a) Debug through UART0 should be displayed through Putty Serial
    b) “Wifi Connected”, “IP” should display through Putty Serial
    c) Connection details should display through Putty Serial when requesting resources through browser
    d) JSON resources should be available through browser

Testing Results



JSON results from requesting resource at IP address of PICO (ESP01)

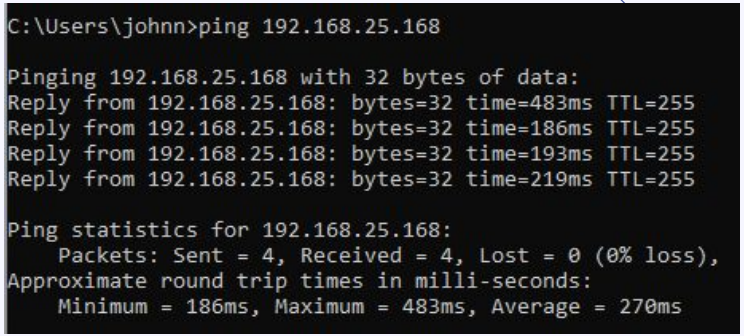
Putty showing resources being requested

Performance

Parameters:
ESP01 connected through a router
Target Device -> Router (Less than 1m)
Router -> ESP01 (Less than 1m)

Average Latency (Calculated by taking average of response time):
(483 + 186 + 193 + 219) / 4 = 270.25 ms

Throughput: ~6Mbps
Max Range: ~200m



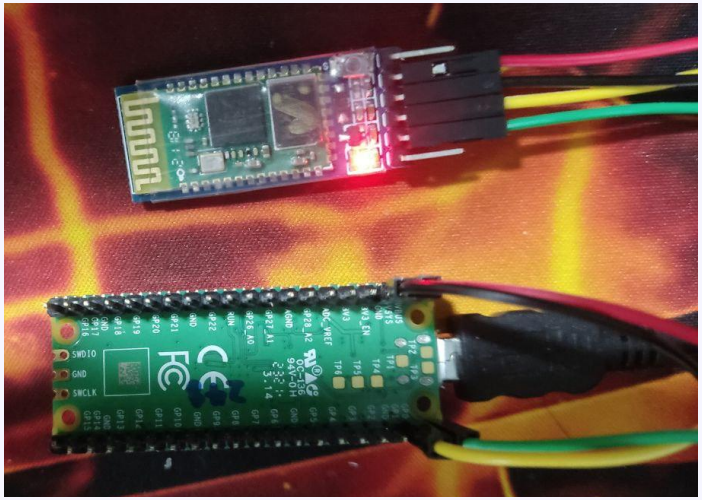
Using CMD ping to measure latency

Demo

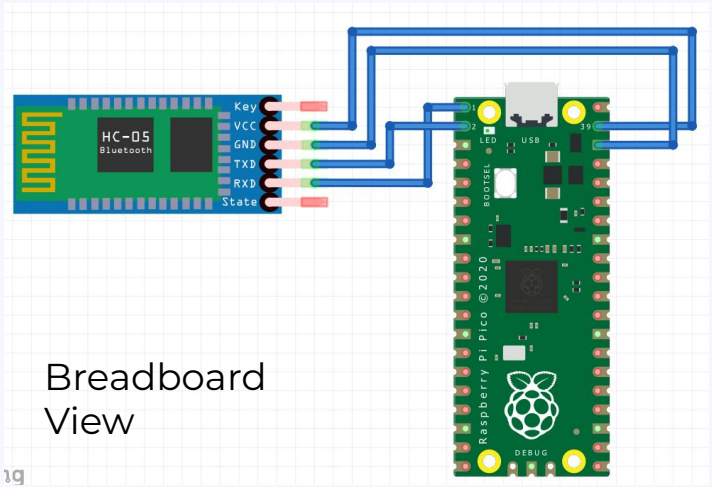
04

Bluetooth

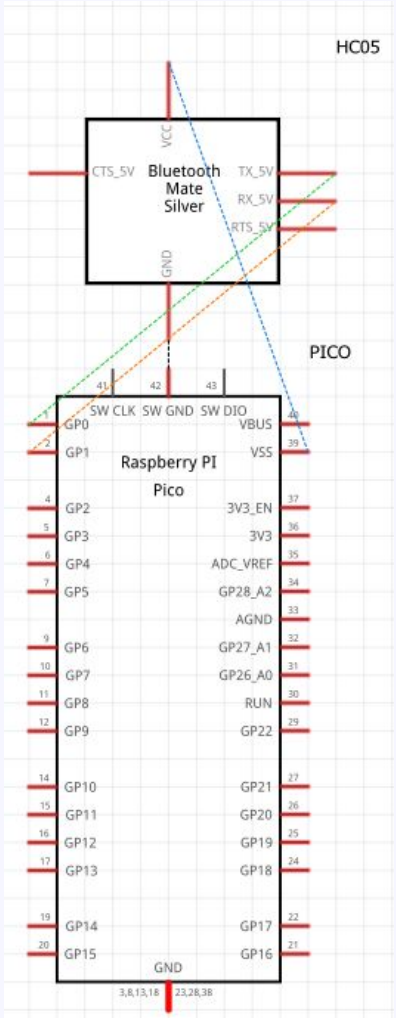
Setup



Actual Setup



Breadboard View



Schematic View

Config

Pico Config-
UART0
TX_PIN 0, RX_PIN 1
Baud Rate 9600
Databits 8 | Stopbits 1 |Parity None

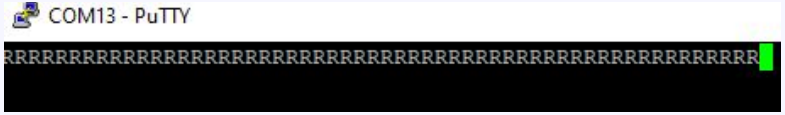
Pin Connection-
Pico GP0 (TX) -> HC05 (RX)
Pico GP1 (RX) -> HC05 (TX)
Pico VSYS -> HC05 (VCC)
Pico GND -> HC05 (GND)

Testing

Conducted black box testing

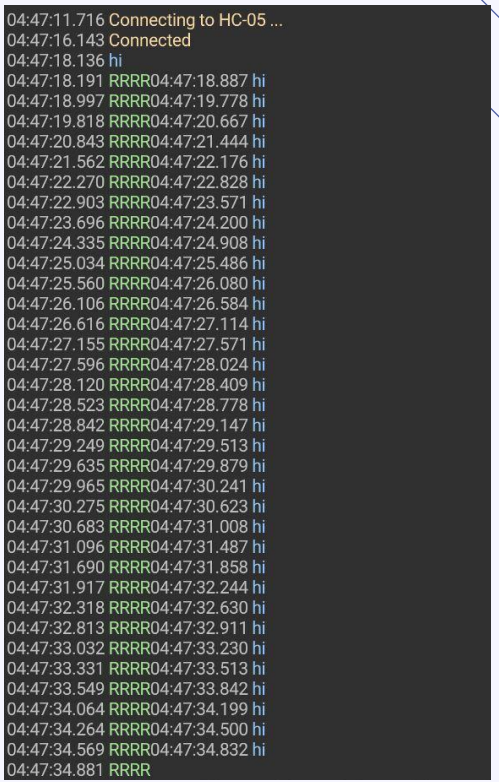
1) Testing Criteria
    a) ‘HC05’ should show up in bluetooth devices
    b) ‘HC05’ should be connectable through bluetooth
    c) Serial comms should be avail once bluetooth connected and able to connect
    d) Output should display ‘R’ (Sent by Pico) everytime something is inputted from Device to Pico

Testing Results



Serial connected through Bluetooth using Putty

- Shows the ‘R’ being returned on every keystroke



Serial Bluetooth Terminal (Android)

Troubleshooting

1) Hold BOOTSEL (black button) on HC05 and connect to VSYS to enter BOOT MODE (Slow red flash every 2s)
2) Send the following command through PICO UART to HC05 to restore default settings
3) Reconnect HC05

4. Restore default

Command	Respond	Parameter
AT+ORGL	OK	-

Default state:

Slave mode, pin code :1234, device name: H-C-2010-06-01 ,Baud 38400bits/s.

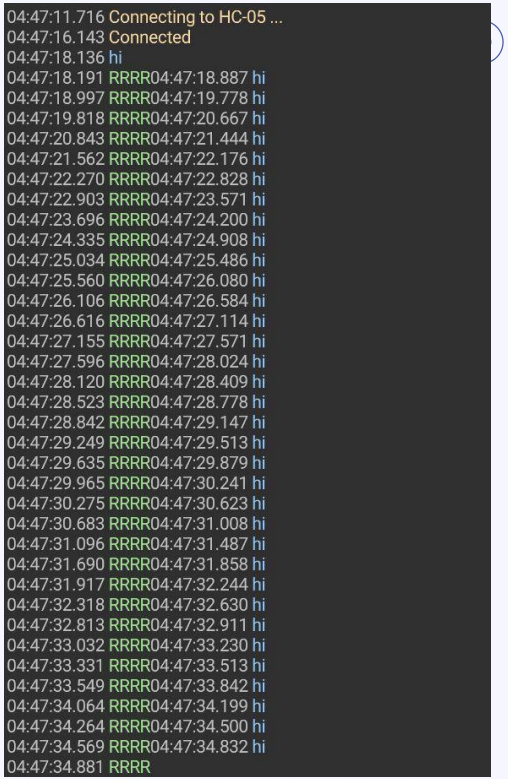
Performance

Parameters:
Target device is placed within 10cm of HC05

Average Latency (Calculated by taking average of response time):
[(191-136) + (997-887) + (818-778) + (843-667) + (562-444)] / 5 = 99.8ms

Throughput: 2.1Mbps(Max) / 160 kbps

Max Range: ~10m



Using Serial Bluetooth Terminal for testing

Demo

Comparison

	Latency (in ms)	MAX Throughput (in MB/s)	Range (in m)
HC05 via UART	99.8	2.1	10
ESP01 via UART	270.25	6	200
M5Stick via UART	250	20	250

Chose ESP01:

- Lightest weight (Car already very heavy)
- High Range
- Enough Throughput (Only sending JSON)
- Acceptable Latency

Thank You!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon** and infographics & images by **Freepik**