11TH Mar

**3RD PHASE PRESENTATION**

Working Prototype

**Bluetooth module,**

INTRODUCTION HC‐05 module is an easy-to-use Bluetooth SPP (Serial Port Protocol - In computing, a serial port is **a serial communication interface through which information transfers in or out sequentially one bit at a time**).

This is in contrast to a parallel port, which communicates multiple bits simultaneously in parallel. module, designed for transparent wireless serial connection setup. The HC-05 Bluetooth Module can be used in a Master or Slave configuration, making it a great solution for wireless communication. This serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04‐External single chip Bluetooth system with CMOS technology and with AFH (Adaptive Frequency Hopping Feature). Bluetooth module HC-05 The Bluetooth module HC-05 is a MASTER/SLAVE module. By default the factory setting is SLAVE. The Role of the module (Master or Slave) can be configured only by AT COMMANDS. The slave modules cannot initiate a connection to another Bluetooth device, but can accept connections. Master module can initiate a connection to other devices. The user can use it simply for a serial port replacement to establish connection between MCU and GPS, PC to your embedded project, etc. Just go through the datasheet for more details.

Bluetooth Terminal HC-05

How to use

Open Bluetooth Terminal app. From the menu, tap on "Connect a device - Insecure". You will see a pop-up of "Paired Devices", tap on "HC-05", after a second you will get a toast notifying "Connected to HC-05". Now type "a" and send, the LED will turn on, similarly, send "b" to turn the LED off.

Hardware aspects (excluding raspberry pi and camera feed module)

, Robot control (working)+ related scripts

import RPi.GPIO as GPIO

import serial

temp = serial.Serial("/dev/ttyS0",9600, timeout = 0.1)

m1a = 5

m1b = 6

m2a = 13

m2b = 19

data = ""

fr = "f"

bc = "b"

lt = "l"

rt = "r"

st = "s"

def forward():

GPIO.output(m1a, True) #

GPIO.output(m1b, False) #

GPIO.output(m2a, True) #

GPIO.output(m2b, False) #

def backward():

GPIO.output(m1a, False) #

GPIO.output(m1b, True) #

GPIO.output(m2a, False) #

GPIO.output(m2b, True) #

def left():

GPIO.output(m1a, False) #

GPIO.output(m1b, True) #

GPIO.output(m2a, True) #

GPIO.output(m2b, False) #

def right():

GPIO.output(m1a, True) #

GPIO.output(m1b, False) #

GPIO.output(m2a, False) #

GPIO.output(m2b, True) #

def stop():

GPIO.output(m1a, False) #

GPIO.output(m1b, False) #

GPIO.output(m2a, False) #

GPIO.output(m2b, False) #

def main():

# Main program block

mon = 0

print ("Welcome to project...")

GPIO.setmode(GPIO.BCM) # Use BCM GPIO numbers

GPIO.setup(m1a, GPIO.OUT) # motor pin1

GPIO.setup(m1b, GPIO.OUT) # motor pin1

GPIO.setup(m2a, GPIO.OUT) # motor pin1

GPIO.setup(m2b, GPIO.OUT) # motor pin1

stop();

time.sleep(1) # 3 second delay

while True:

data = temp.read()

#print(data)

if data.rfind(fr) > -1:

print ("Forward")

forward();

elif data.rfind(bc) > -1:

print ("Backward")

backward();

elif data.rfind(lt) > -1:

print ("left")

left();

elif data.rfind(rt) > -1:

print ("right")

right();

elif data.rfind(st) > -1:

print ("stop")

stop();

if \_\_name\_\_ == '\_\_main\_\_':

try:

main()

except KeyboardInterrupt:

pass

finally:

GPIO.cleanup()