Object Avoidance Robot

Harshil Bhojwani
CS7680 Special Topics: Programming Embedded System
Roux Institute, Northeastern University
Portland, Maine
bhojwani.h@northeastern.edu

I. Introduction

Designing a vehicular robot is an excellent entry-level project to gain a sound understanding of hardware design and explore various communication protocols. It provides practical insights into how different hardware components and protocols can work together cohesively, enabling the development of a fully functional system.

In this project, I developed a Raspberry Pi-based robot with an additional microcontroller (MCU) to explore key concepts such as working with registers, implementing I2C communication, and performing analog-to-digital conversion. These elements, combined with the use of PWM inputs, allowed me to create a fully functional robot.

The robot was equipped with various hardware components, including DC motors, a servo, a proximity sensor, and a grayscale sensor. With this setup, I successfully designed a robot capable of avoiding obstacles and detecting cliffs, demonstrating effective integration of hardware and software.

II. Design

A lot of my design was restricted considering the robot chasis was not my design with instructions of fitting each set of equipment at its required place, even the MCU which is connected to the rasperberri pi already had , as for an embedded system engineer there are few instances where they are given the freedom to make their own design rather are been given the responsibility of executing other design with which made this project a great experiance as I had to be creative withing the realms of hardware constraints I was challenged with

Below are the definitions of few protocols which I came across and used in this project

I2C :- I2C stands for Inter-Integrated Circuit. It is a bus interface connection protocol incorporated into devices for serial communication. It is widely used for attaching lower-speed peripheral integrated circuits (ICs) to processors and microcontrollers in short-distance, intra-board communication. where one device acts as a master controlling communication with other devices acting as slaves; it utilizes two lines, one for data (SDA) and one for clock (SCL) to transfer information between devices. Each slave device has a unique address to identify it on the bus.

ADC :- Considering the raspberri pi by default only reads Digital signals and ADC helps as it converts converts an analog signal (continuous in time and amplitude) into a digital signal (discrete in time and amplitude). ADCs are widely used in applications like sensors, audio processing, and data acquisition systems where real-world analog signals must be processed digitally.

PWM :- PWM, or Pulse Width Modulation, is a technique used to control the power delivered to electrical devices by varying the width (duration) of pulses in a signal. PWM is commonly used in applications like motor speed control, LED dimming, and communication systems. PWMs are extremely useful in controlling the motor speed.

Below are the different hardware components which were used for this project.

TT motor A TT motor is a type of DC motor that has a gearbox attached to it. The gearbox reduces the speed of the motor and increases its torque. A TT motor is commonly used in applications such as driving wheels, propellers, fans, among others. A TT motor has two wires: a positive wire and a negative wire. The positive wire is usually red and the negative wire is usually black.

Fig. 1. TT motor

Servo motor Servo motors are devices that can rotate to a specific angle or position. They can be used to move robotic arms, steering wheels, camera gimbals, etc. Servo motors have three wires: power, ground and signal. A servo is generally composed of the following parts: case, shaft, gear system, potentiometer, DC motor, and embedded board.

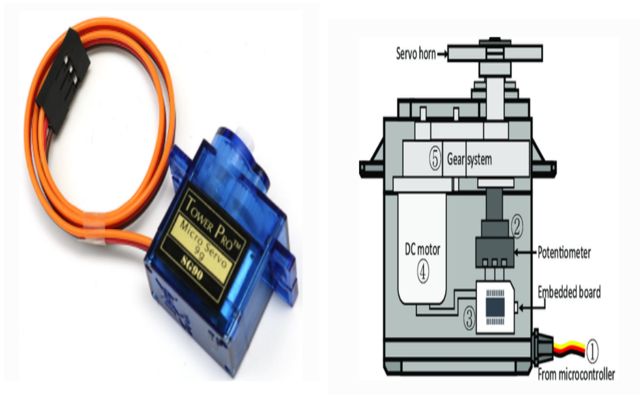


Fig. 2. (a) Servo (b) Servo Components

GreyScale sensor grayscale sensor module consisting of 3 TCRT5000 transmitting sensors, which can be used for line following and edge detection. The TCRT5000 transmitter sensor consists of an infrared light-emitting diode and a phototransistor covered with a lead material to block visible light. When working, the IR LED of TCRT5000 continuously emits infrared light (invisible light) with a wavelength of 950nm. When the emitted infrared light is not reflected back by the obstacle or the reflection intensity is insufficient, the phototransistor does not work. When the infrared light is reflected with sufficient intensity and received by the phototransistor at the same time, the phototransistor is in working condition and provides output. In general, white surface return value $>$ black surface's $>$ cliff's. Also the brighter the surface, the more reflections, resulting in an increase in current, the brighter the LED.

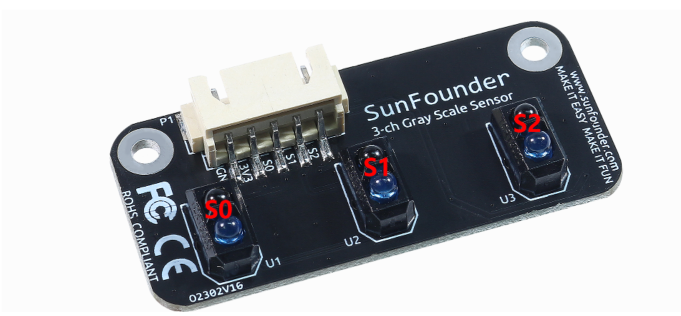


Fig. 3. Grey Scale Module

MCU Robot HAT is a multifunctional expansion board that allows Raspberry Pi to be quickly turned into a robot. An MCU is on board to extend the PWM output and ADC input for the Raspberry Pi, as well as a motor driver chip, Bluetooth module, I2S audio module and mono speaker. As well as the GPIOs that lead out of the Raspberry Pi itself. The Robot HAT comes with an AT32F413CBT7 microcontroller from Artery. It is an ARM Cortex-M4 processor with a maximum clock frequency of 200MHz. The microcontroller has 128KB of Flash memory and 32KB of SRAM.

The onboard PWM and ADC are driven by the microcontroller. Communication between the Raspberry Pi and the microcontroller is established via the I2C interface.(7 bit address format)

III. Build Process

A. *TT motor*

I should start explaining how I brought together and assembled the robot but as mentioned earlier that I had not effort towards designig the chasis it is something which I directly picked it up form the market rather I only spent effort in prgoramming it from my own code.

My first step was to get the robot moving, as every TT motor is attached to two GPIO pins 1 of those pins are for direction pins which is moving the motor i forward or reverse direction **1** indicates the robot to move forward and **0** indicates the robot to move in reverse direction. The second gpio connection for the DC helps to set the speed at which the TT motor will rotate through **PWM** the TT motor functions at a speed of 1KHz with a speed to be set between **0 - 100**.

Using functions available in **PIGPIO** library in **C** make this task fairly easy to program this, though it is important to read the specifications of hardware being it the speed and the range to program it accordingly, more will be spoken about those in experimentation.

B. *Proximity Sensor*

Proximity sensor consist of an ultrasonic reciever and transmitter within it, as the front servo helping to control the direction of the robot are dependant on the detection of these sensors, the second step was to program these.

The transmitter and reciever are connected to 1 GPIO pin each where the reciever is an input device and the transmitter is an output device. The direction is only suppose to change when the sensor receiving the signal outputs a distace which is too close for the robot to collide.

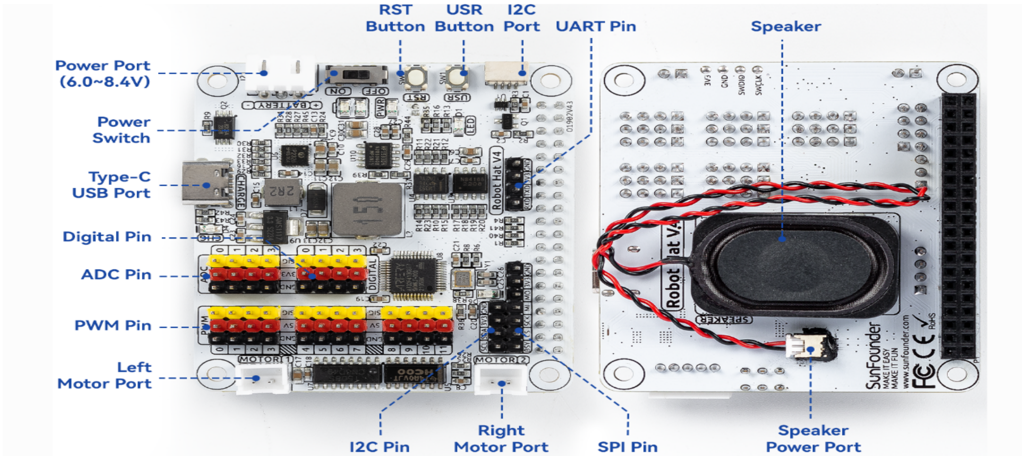
The transmitter continuously sends signals in the form of ultrasonic waves at a frequency of 40KHz and the receiver checks if they receive any waves in return. The period for which recived remains **High** for is what calculates the distance of the object from the sensor.

Total Distance = High Time x $\frac{soundspeed}{2}$

C. *Grey Scale sensor*

As mentioned earlier, the gray scale sensor helps to detect the cliff, the change in surface which assist in robot to not fall and follow the line, though I am not making the robot follow the line but ensuring that it does not fall of the surface, especially when placed on a desk. These sensor send continuous IR waves and collect the reflection, the IR waves are a from analogue waves and GPIO pins only read digital sensor there they are connected to ADC through I2C to make these sensors synchronously with the Raspberri Pi.

The sensors are connected directly to the **5V** supply on the **ROBOT HAT**, **S0/S1/S2** are the output values of the three transmitting sensors, the output values are analog values.



Below in the experimentation section I will elaborate on how the greyscale sensor was setup.

D. Servo Motor

Servo motor is what controls the front wheels with a job to change its direction, depending on the sensory output from greyscale sensor and the proximity sensor the output is decided, in scenarios when its too close the the edge of the surface or too close the obstacle it the servo will change its direction. The servo is also a form of motor but rather than rotating continuously in this robot they only move to a certain angle to give robot the direction.

The servo are are also powered by PWM but rather then being connected directly to GPIO PIN they are connected via the I2C interface where a prwscaler and period have to be set for them

IV. Experiments

A. ***TT motor***

A TT motor has two pins connected to a PI, one for controlling the Direction which is either it rotates clockwise or anti clockwise, the direction are dictated by setting the GPIO pin mode being set up as 1 which dictates clockwise which is also forward and 0 which dictates anticlockwise also as backward.

The other pin controls the speed as dictated above the motor operates at a Frequency of 1KHz between 0-100 speed, consindering they are connetected to GPIO pins directly, both of the settings can easily be configured using PIGPIO library in C.

The way I configured these motors was to move forward at a 50 percent speed always unless the grey scale sensor detects a cliff and then to only move backward, turn left before again starting to move forward.

B. ***Proximity Sensor***

Proximity Sensor detects the distance from the obstacle by transmitting ultrasonic waves and a receiver collecting those information, as the sensor is connected to the digital pins directly the output for the receiver is a simple 1 and O but time sensor remains at 1 detects the distance of the obstacle from the robot.

The transmitter is configured to give out burst of ultra sonic waves, and the reciever waits to go on High, one i High I record the start time and end time to know the duration.

Once the duration is known the distance can be easily be known by dividing the it by 58.

```
// Calculate distance in cm
duration = endTime - startTime;
distance = duration / 58.0; // Speed of sound conversion to cm
```

Fig. 4. Formula to know the distance from obstacle

C. ***Greyscale sensor***

The greycale sensor was connected to the ADC channels of the Robot-Hat which function via I2C to precisely be able to read the surface reflection in order to detect the surface edge or cliff.

The Robot-Hat has 4 ADC channels, powered by 3 volts each, to which my 3 sensors are connected.

In my experimentation, any return value values less than 10000 usually detected an edge of the surface. Keeping that in mind I programmed my robot, is as soon as greyscale sensor gives a reading less than 10000 a cliff is detected.

PIGPIO Library has several features to enable being to recognize the I2C bus which is active as well and within those to reach the readings of the channels of either ADC or PWM in this case.

After opening the of the I2C bus through PIGPIO built in function, I read 3 byte readings of per ADC channels by creating an array of 3 Bytes and use the bitwise shifting feature in to store and read the readings of every greyscale sensor.

I take the readings from the greyscale sensor every 100 millisecond for a smooth functioning of the system.

D. ***Servo***

The servo is what ensures that the robot does hit an obstacle. Servo on this robot is also connected through PWM channels of the robot-hat via the I2C interface. The MCU has 11 PWM Channels

The servo operates at a frequency of 50Hz capable to rotate at a angles between 0-180. The PWN channels operates at a CPU clock speed of 72 MHZ.

Register from I2C interface is to set the PWM prescaler. ranges from 0 65535. There are only 4 timers for all 14 channels.

Register from I2C interface is to set the PWM period. ranges from 0 65535. There are only 4 timers for all 14 channels.

PWM Timer is a tool to turn on and off the PWM channel for you.

The MCU only have 4 timers for PWM: which means you cannot set frequency on different channels at with the same timer.

Example: if you set frequency on channel 0, channel 1, 2, 3 will be affected. If you change channel 2 frequency, channel 0, 1, 3 will be override.

In realife the stright Position of the servo is 90 degrees whenever it detects an obstacle which is less than 10cm it always turns left which is set at 180 degrees.

FUTURE WORK

In Future this robot comes with two more functionality where a camera can be attached to it, the idea is to further build upon this and and be able to integrate a camera inorder to have some latest computer vision object detection algorithm be running on it and with a speaker built inside the Robot Hat, I would want to configure it to give out timely prompts when an edge or obstacle is detected.

ACKNOWLEDGMENT

I would want to thank Dr Scott Valcourt for his CS7680 Special Topics in embedded System Class inorder to achieve a lot of these accomplishments in this project.