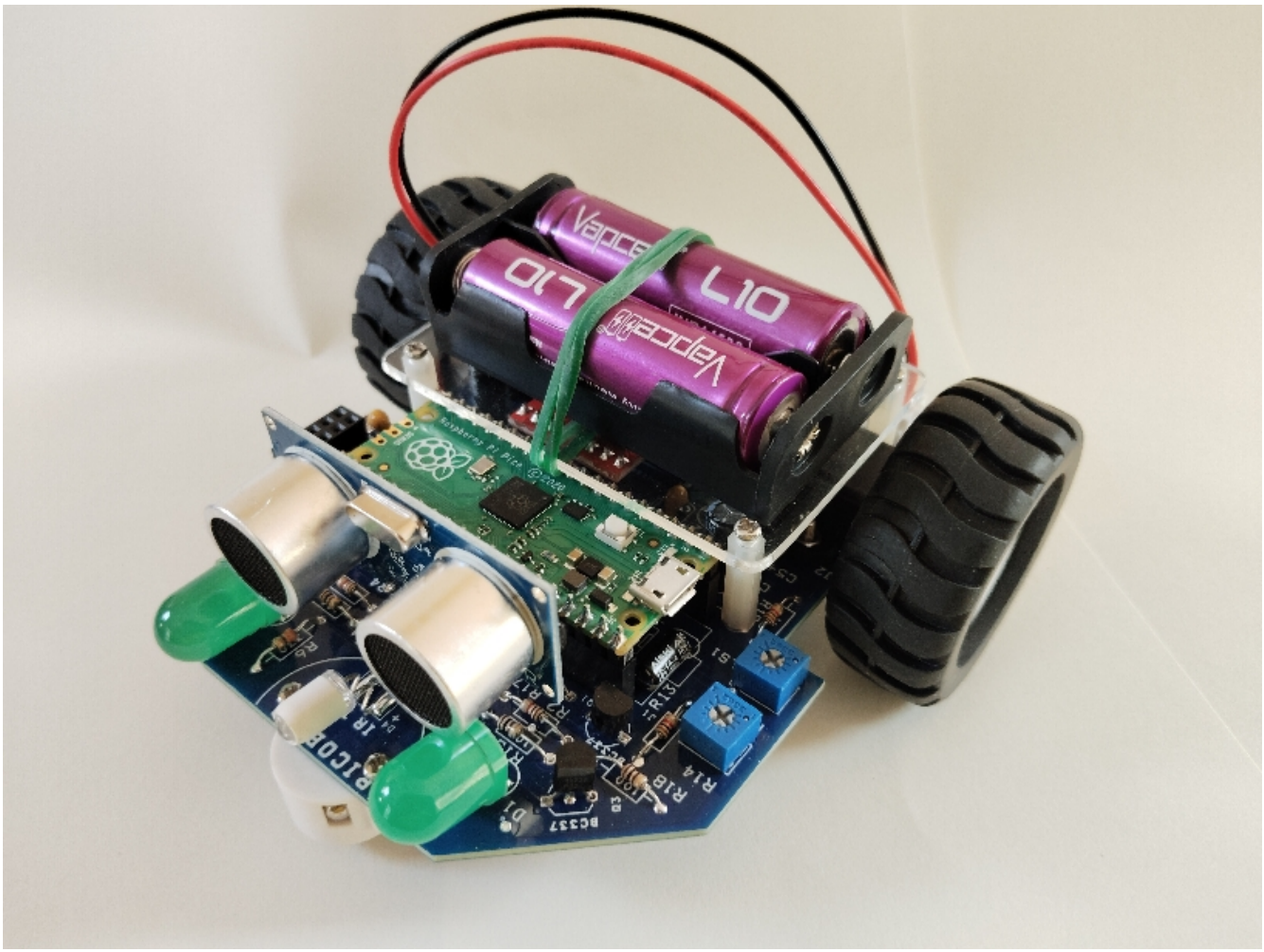
PICOBOT v1.0

LAB BOOK

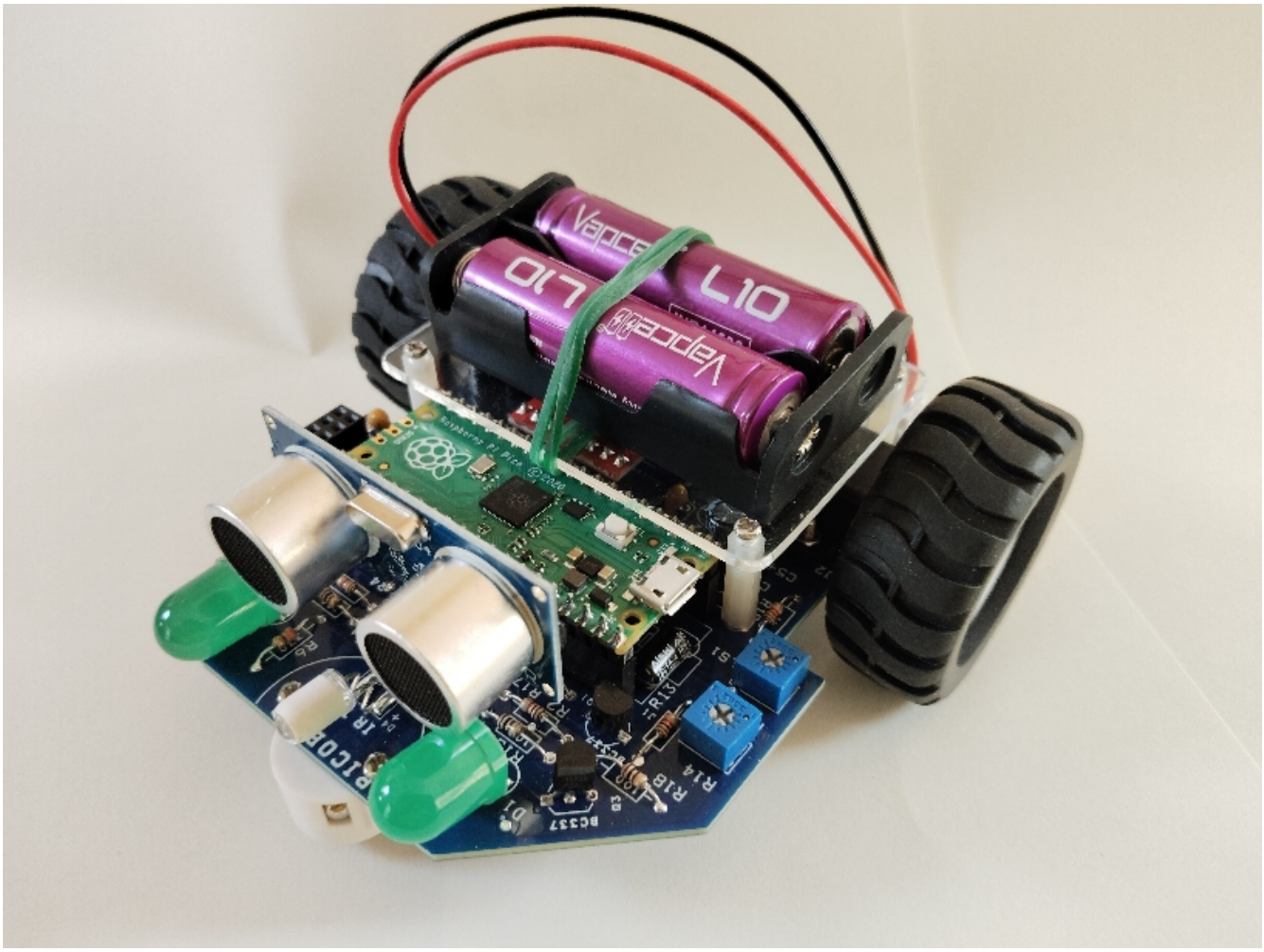


Rev 1.0 November 2025

CONTENTS

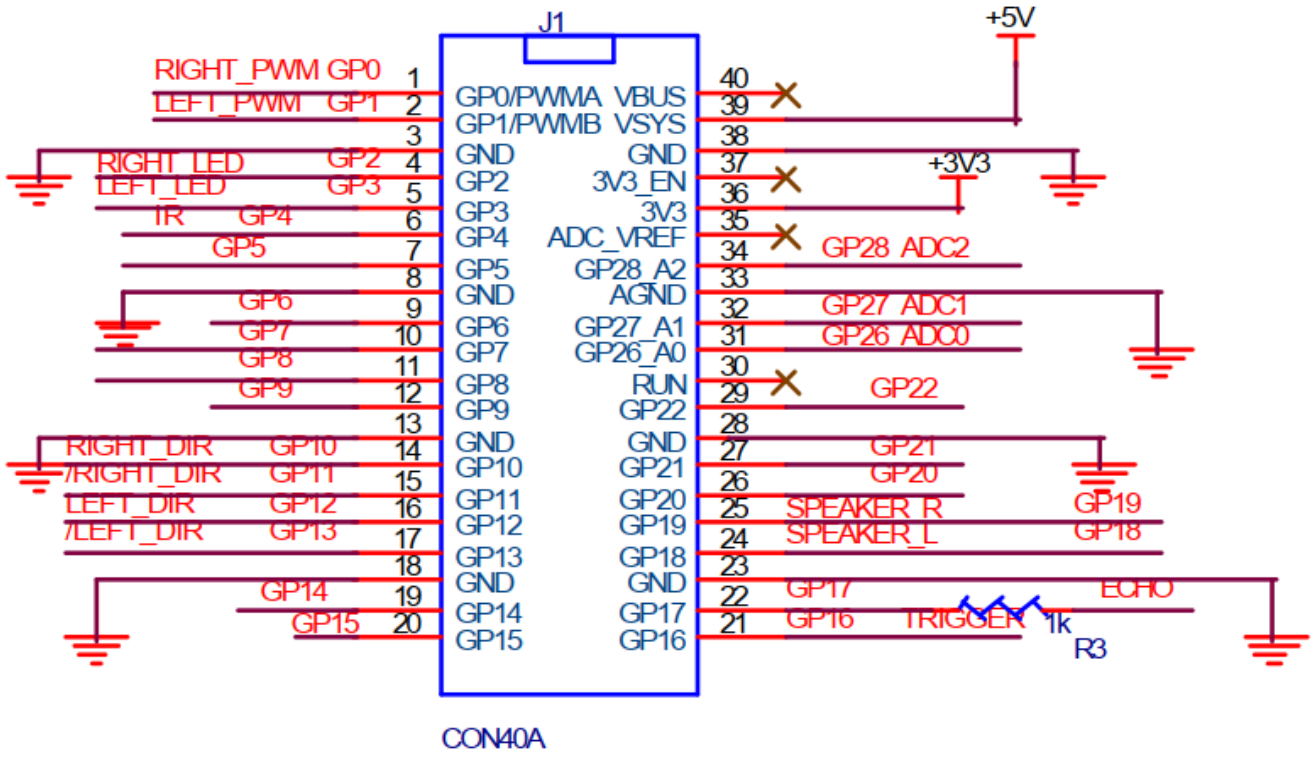
Introduction

The PicoBot v1.0 is an educational robot programmed by a MMBasic interpreter. The base controller is a PI Pico compatible board. The built-in sensors are an ultrasonic ranger, line tracking sensors, and an IR receiver/transmitter. The bot also has two headlights, two audio channels, and Start/Stop buttons. The bot connects to the serial terminal for basic program writing. The interpreter allows command mode and program mode testing. In command mode, we can write a subroutine that controls the motor. Then we can test motor running by the name of a subroutine under command mode directly. For program running mode, the option AUTORUN, when set, will let the bot run the program automatically.



I/O port signals

The schematic diagram shows a 40-pin I/O port of the PI pico RP2040 board. These I/O ports are connected to the sensors and motor driver. The on-board USB port is for connecting to the serial terminal.



LAB 1 Control the Headlight using command mode

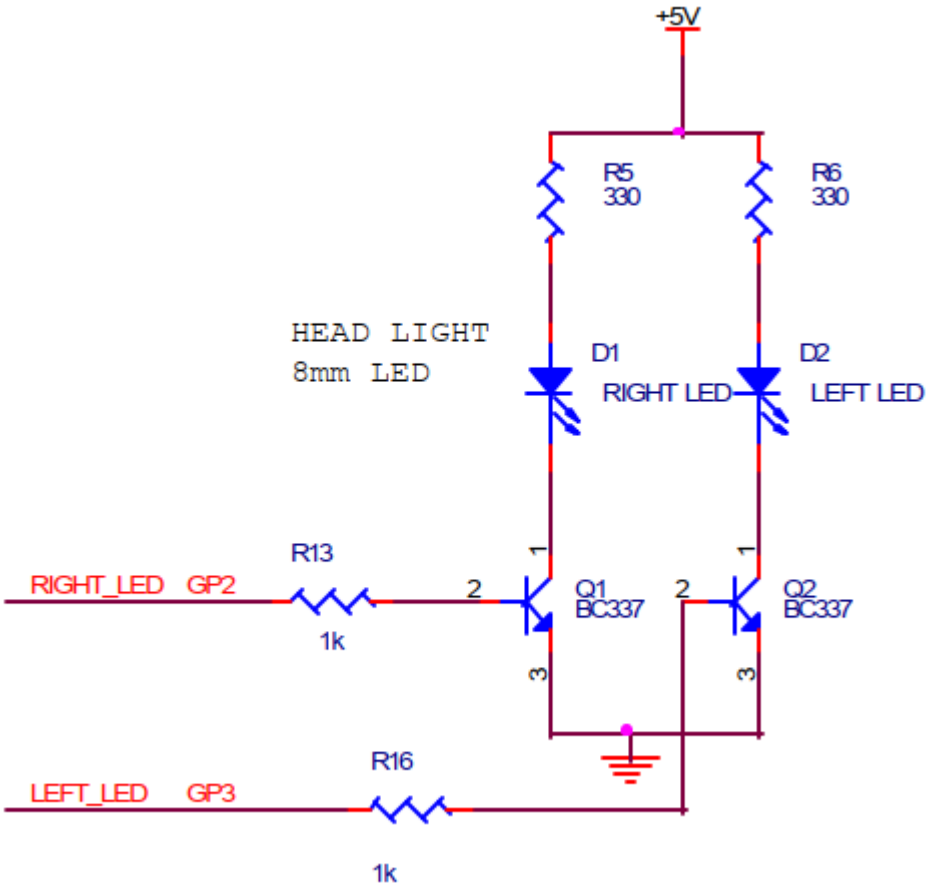
1. Demonstrate how to use MMBasic in command mode

2. Basic setting for controlling the I/O pins

The bot has two headlights, left and right 8mm LEDs.

The right LED is connected to GP2, the left LED to GP3. These two bits can be set to be digital output using the Setpin command.

Setpin GP2, DOUT
Setpin GP3,DOUT



To set logic 1 or 0 at these ports, use command Pin(GP2) = 1 or Pin(GP2)=0

Under command prompt> type below command

```
>
> Setpin GP2,DOUT
> Pin(GP2)=1
>
```

To turn on the LED, set the port bit to logic 1 with command Pin(GP2)=1

Can you turn it off ? How?

Similar to GP3, to turn the LED, use command Pin(GP3)=1

```
>
> Setpin GP3,DOUT
> Pin(GP3)=1
>
```

Can you turn it off? How?

Suppose we want make the headlight as a flash blinking. Under command mode we can use for next control statement as shown below. Type below progam under prompt mode.

When ENTER, we will see the headlight blinking.

```
>
> for i=1 to 10 : Pin(gp2)=1 : Pause(100) : Pin(gp2)=0: Pause(500) :next i
>
```

We set the number of loop to 10. Make the logic at GP2 to 1 then pause 100ms, then make it to logic 0, pause 500ms. Next I will repeat the loop 10 times.

Can you change the speed of blinking? How?

Headlight LED is very useful for program debugging. We can use it as the indicator for program flowing, check the bot functioning easily.

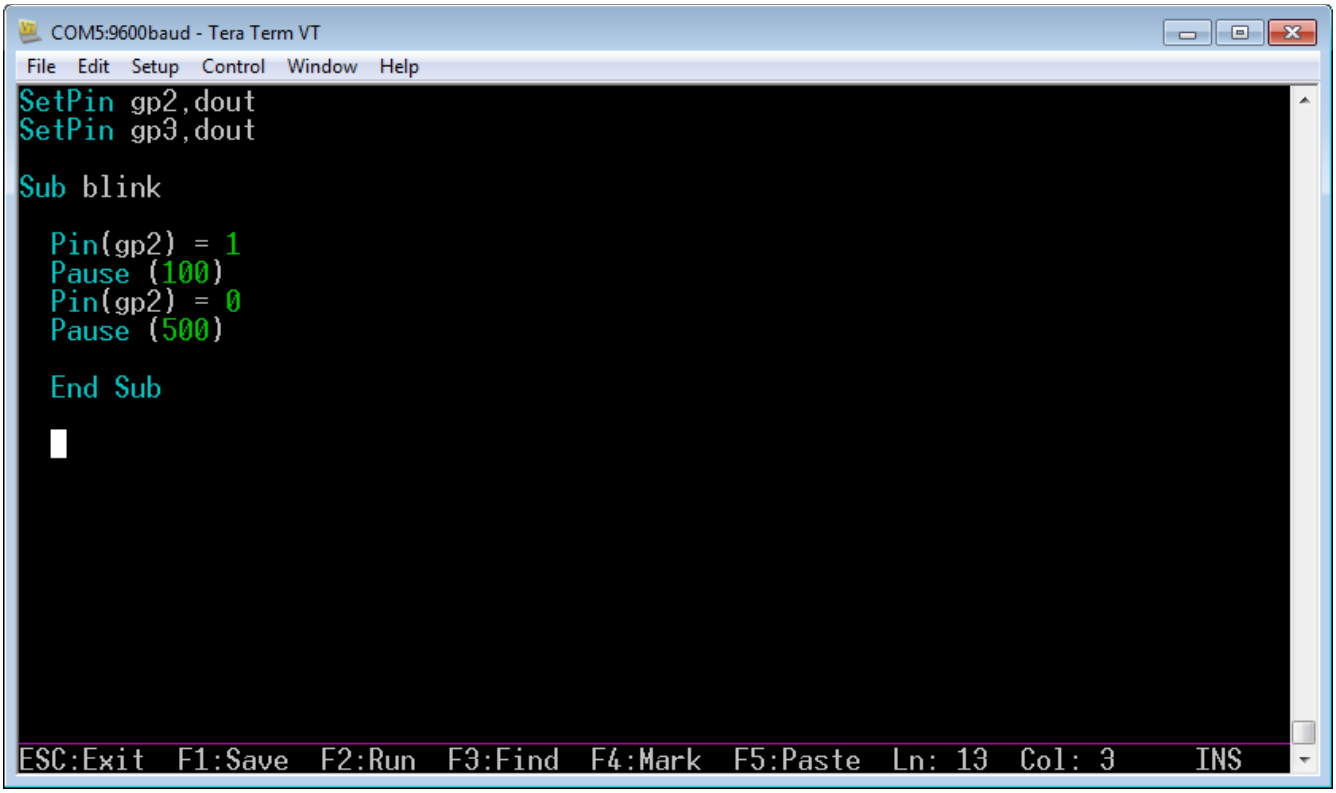
LAB 2 Subroutine and Built-in editor

1. Writing subroutine
2. Using built-in Editor

The MMBasic provides built-in Editor program. Under prompt mode, type edit, to run editor.

With editor, we can write a long program. This Lab will show how to write the subroutine.

Suppose we want to make a blink subroutine, we can type below. The subroutine name is Blink. This subroutine will make GP2 logic to 1, pause 100ms, then clear it to 0, pause(500).



Here is the list of subroutine blink.

```
SetPin gp2,dout
SetPin gp3,dout

Sub blink

  Pin(gp2) = 1
  Pause (100)
  Pin(gp2) = 0
  Pause (500)

  End Sub
```

When complete, press F1 to save the program and exit.

```
Saved 134 bytes
> Blink
> blink
>
```

Try with new command, blink.

Can you make both headlight blinking? How?

We can use for loop the same as LAB1 for making a number of blinking easily.

```
>
> for i=0 to 10: blink : next i
>
```

Can you change the number of loops? How?

Now if we use do loop statement, what is happening?

```
>
> do : blink : loop
```

```
>
```

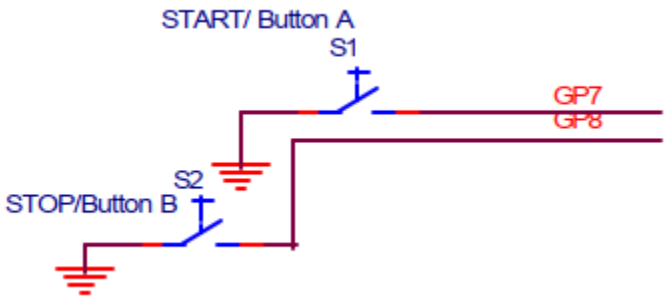
Blinking will be repeated. To exit the running program, use Crtl C.

LAB 3 Testing Start/Stop button

1. Test the logic reading from the Start/Stop button

2. Control the headlights with the Start/Stop button

The bot provides user buttons, Start and Stop buttons for many experiments. We can use such a button for controlling the bot.



Start Button A is connected to GP7.

Stop Button B is connected to GP8.

To set a pin for digital input with pullup, we can use the command

Setpin GP7, DIN, pullup

When the button contact is opened, GP7 will be logic 1. When we press the button, close to GND, the logic at GP7 will be 0.

We can use the logic 1 or 0 for controlling the bot then.

Let us test reading the logic at GP7, START button under command mode.

```
> setpin gp7, din, pullup
> print pin(gp7)
 1
> print pin(gp7)
 1
> print pin(gp7)
```

```
 0
> print pin(gp7)
 1
>
```

We set GP7 to be digital input with pullup using the command Setpin gp7, din, pullup.

When we use the command print pin(gp7) we will see logic 1 if no press. And logic 0 when the Start button is pressed. To repeat the current command under prompt, MMBasic uses key Arrow.

Can you do the same using the Stop button at the GP8 pin? How?

Now get back to the editor, add the code to the subroutine blink.

```
SetPin gp2,dout
SetPin gp3,dout

Sub blink

  Pin(gp2) = 1
  Pause (100)
  Pin(gp2) = 0
  Pause (500)

  End Sub

SetPin gp7,din,pullup
SetPin gp8,din,pullup

Do

  If Pin(gp7)=0 Then blink
  Pause (100)

  Loop
```

When complete, press F1 to save the program and exit at the prompt.

We add the code Do loop, to repeat reading the logic at GP7. If the logic at GP7 is zero, then call the subroutine blink.

When we run the program, >RUN, then press the Start button, the bot will blink in the headlights.

Can you add the code for checking the Stop button and make some blink when the Stop button is pressed? How?

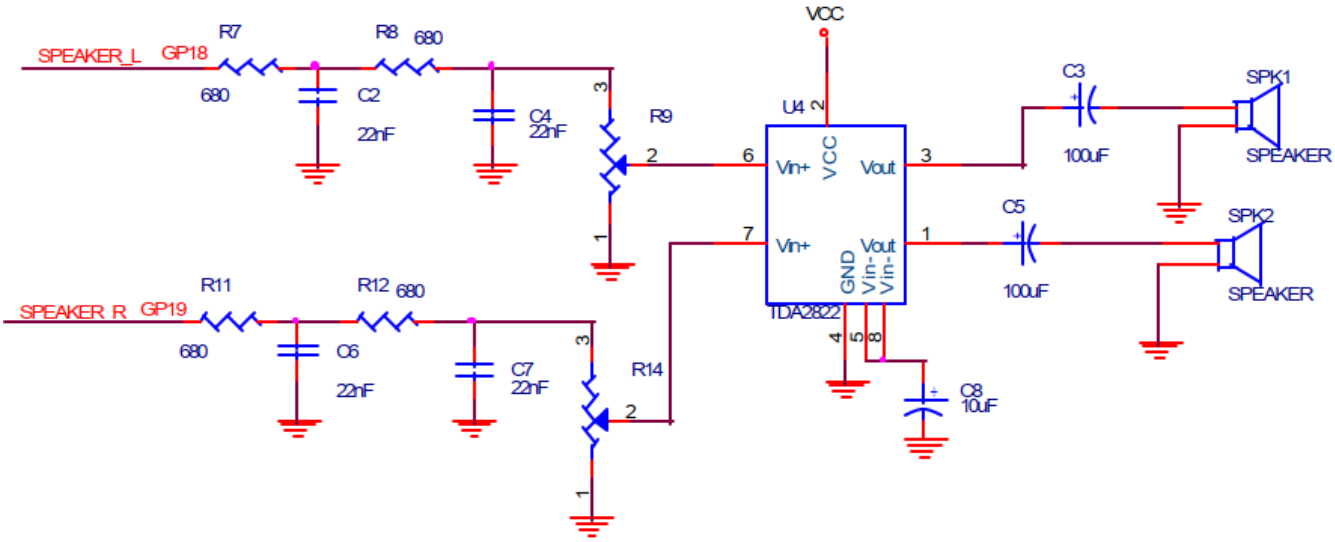
LAB 4 Testing play audio

1. Producing tone with play command

2. Making horn controlled with the Start button

The bot provides stereo audio output channels left and right. The audio signal is PWM with a 44kHz carrier frequency. A simple 2nd order low-pass filter using RC is used to remove the carrier signal. The audio signal is then amplified with a TDA2822 stereo amplifier.

We can test the code for producing a tone signal from a BASIC program directly.



MMBasic provides an option command for setting the PWM channel to be audio output.

To check the hardware settings, we can type the command OPTION list.

```
>
> option list
PicoMite MMBasic RP2040 Edition V6.00.03
OPTION AUTORUN  ON
OPTION COLOURCODE ON
OPTION CPUSPEED (KHz) 200000
OPTION AUDIO GP18,GP19', ON PWM CHANNEL 1
>
```

The PWM channel1 is used for audio signal generation.
Let us try producing the horn manually.

Channel A is 500Hz and Channel B is 455Hz.

```
>
> play tone 500,455
>
```

To stop it, press Crtl-C or command Play tone 0,0

Can you change the tone frequency? How?

Suppose we want to use Start button A to be a horn control, how to do that?

Edit the program below using the Editor.

```
SetPin gp7, din, pullup

Do

 If Pin(gp7)=0 Then Play tone 500,455
 If Pin(gp7)= 1 Then Play tone 0,0

  Loop
```

The program keeps reading the logic at pin GP7. When we press the Start button A, function play tone will be activated.

If we release, GP7 will be logic 1, then tone frequency will be 0,0.

Can we use the START button to start the sound and use the STOP button to stop it? How?

LAB 5 Using IR remote control

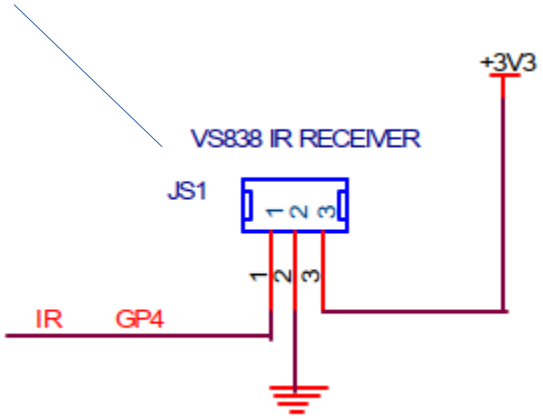
1. Detecting the key code from IR remote control

2. Making key code map

The bot has an IR receiver placed on the rear side. MMBasic detects codes using 12-bit Sony format. MMBasic can also detect code from a cheap remote control. The code may not the same as Sony code, but we can adapt it for many experiments.

The detected pulse signal is connected to GP4.

IR receiver



Remote control

To use GP4 as an IR receiver port, we can use the command Setpin gp4, IR

Try editing the program below using the Editor.

```
SetPin gp4, ir
Dim integer devcode, keycode
IR devcode, keycode, irint

Do
  ' main body
 Loop
```

```
Sub irint
 Print devcode, keycode
 End Sub
```

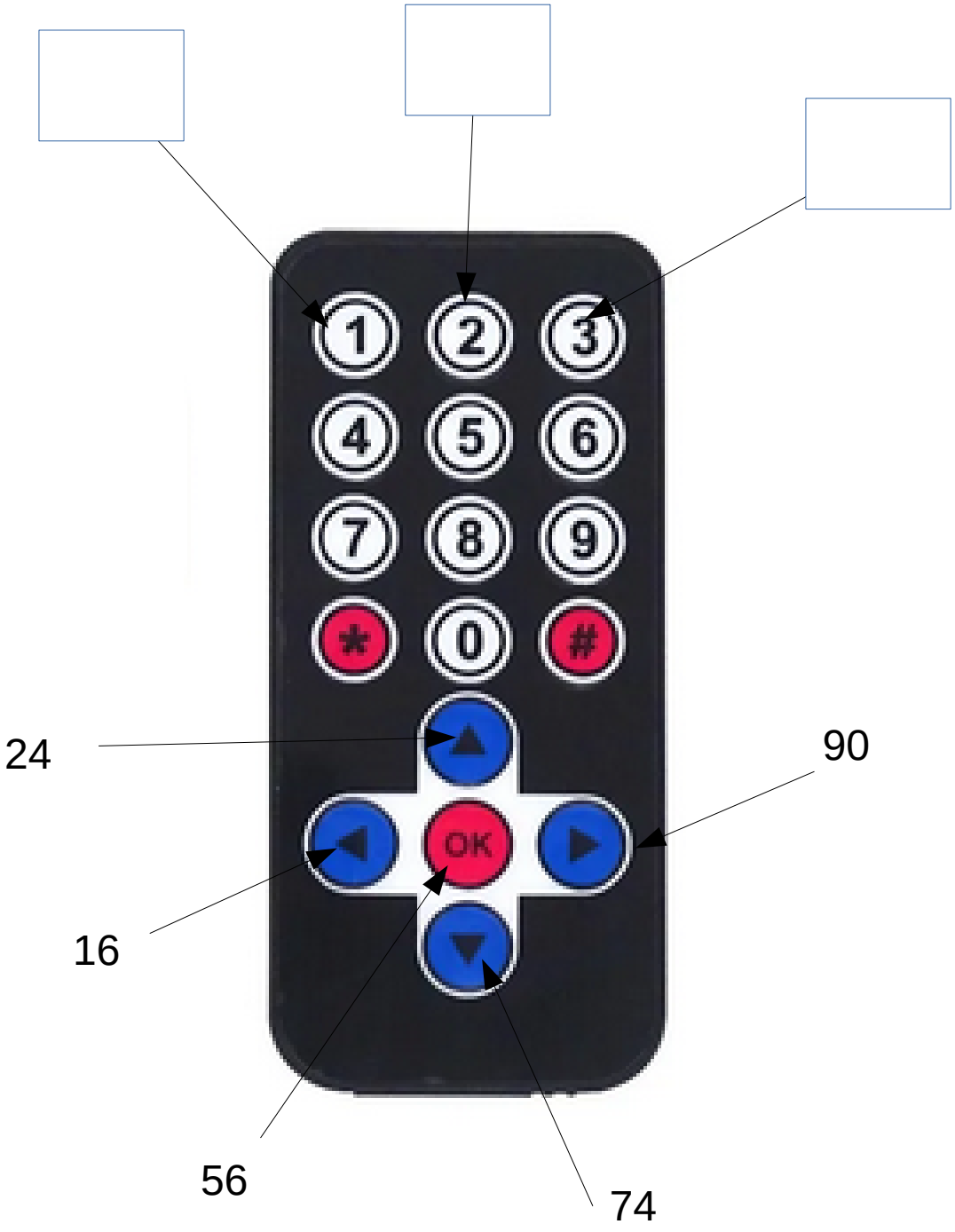
We set pin gp4 to set the IR receiver pin. The receiver function is operated by IR interrupt.

When the pulse signal from remote control is detected, the IR interrupt subroutine will print the device code and key code.

When we run the program, press a key on a remote control, the program will print the device code and key code.

```
> run
 255     56
 255     24
 255     90
 255     16
 255     74
 255     104
 255     176
 255     224
 255     152
 255     168
 255     144
```

Can you write down a key code for each key?



From LAB4, we use the START button, when pressed, it will send a BEEP. If we use this remote control to make such a beep, how to do that?

```
SetPin gp4, ir
Dim integer devcode, keycode
IR devcode, keycode, irint

Do
 ‘ main body
 Loop

 Sub irint

  Print devcode, keycode
   If keycode=176 Then Play tone 500,415,200

 End Sub
```

We use key #, code 176, now we can control the beep using remote control.

What about flashing the headlights when we press the key *?

```
SetPin gp4, ir
Dim integer devcode, keycode
IR devcode, keycode, irint



SetPin gp2,dout
SetPin gp3,dout

Do
  ‘main body
 Loop

 Sub blink
  Pin(gp2)=1
  Pause (100)
  Pin(gp2)=0
```

```
 Pause (500)

 End Sub

Sub irint
 Print devcode, keycode

  If keycode=176 Then Play tone 500,415,200
  If keycode=104 Then blink

 End Sub
```

We add the line with key code for * key. And we add the subroutine blink the headlights.

Now run the program.

Use remote control, press key *, or try key #.

The IR remote control can be used for controlling robot movement manually. We will experiment controlling the bot with remote control in LAB 11.

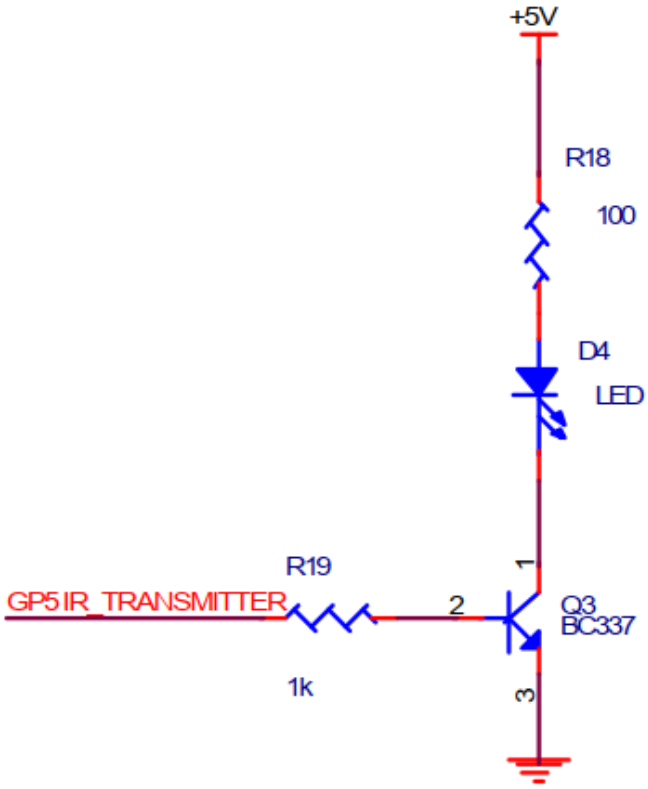
LAB 6 IR Transmitter

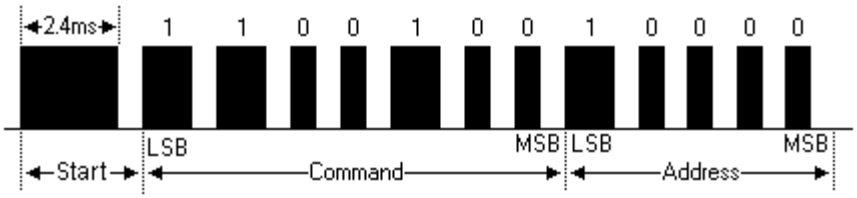
1. Using the command to send IR code

2. Control another bot using an IR transmitter

The bot has a built-in IR transmitter at the front side. MMBasic provides command for sending the code using the 12-Bit Sony Infrared signal. The bot uses GP5 for sending the IR signal. Q3 NPN transistor drives the 5mm IR LED.

The sending command is IR send pin, device, key code.



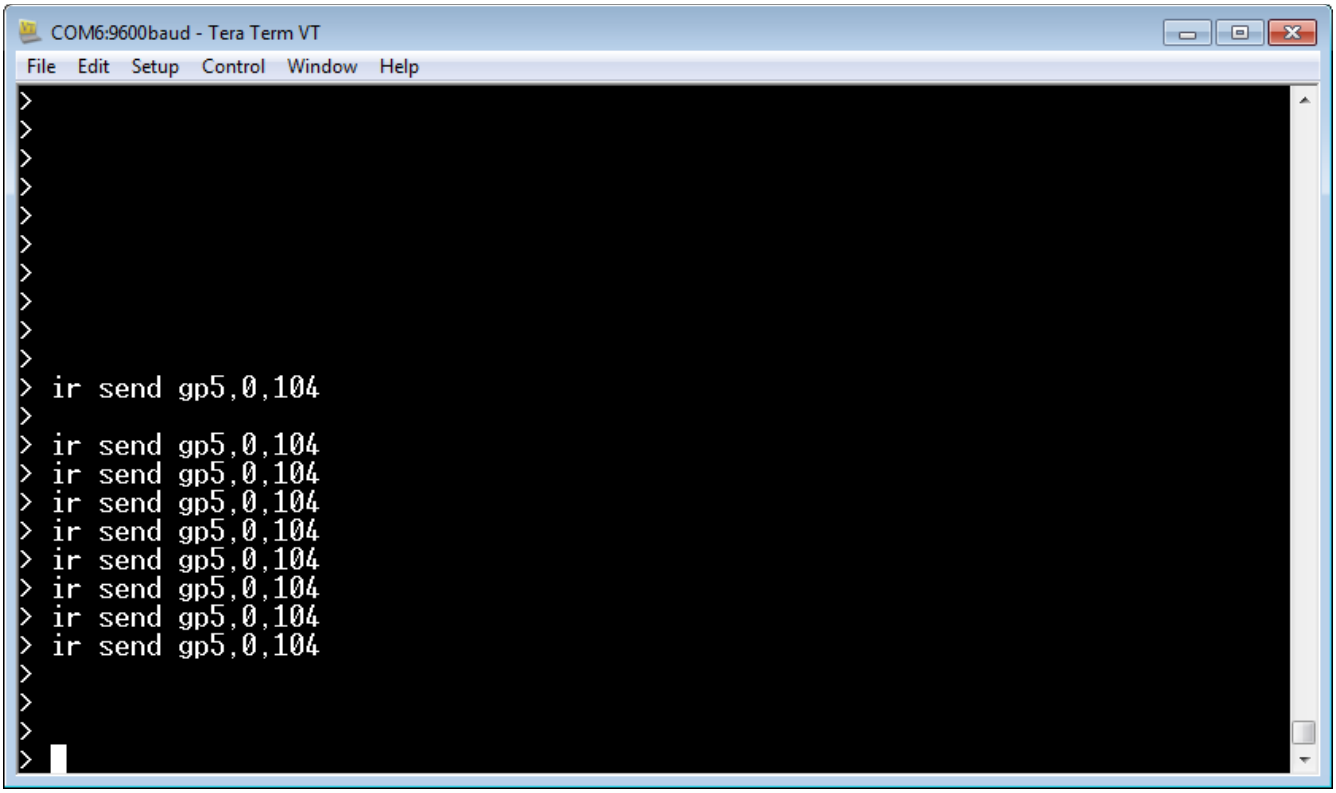


Sony 12-bit command and address. Command is 7 bits, Address is 5-bit, 32 devices.

More information https://www.sbprojects.net/knowledge/ir/sirc.php
We will test send command that blinks the headlight of the second bot using the first bot IR transmitter.

In this test, we will need the second bot runs the program from LAB 5. Place it far from the first bot.

On the first bot, we can simply type command as follows.



Use key Arrow UP to repeat the command. Also, try pointing the IR transmitter in the other direction, towards the room's wall.

Did you get a response from the second bot?

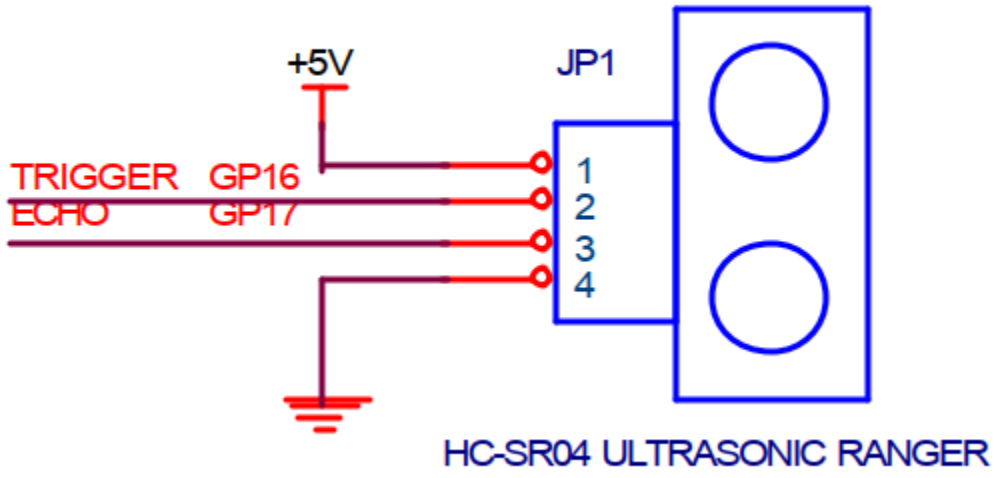
The device can be 0–31 or 32 bots can have its own ID. So, from the program in LAB 5, if we have many bots to control. Each bot can set ID with device 0-31 then.

LAB 7 Ultrasonic ranging module

1. Measure distance with an ultrasonic ranging module

2. Detect the limited range

The ultrasonic ranging module SR04 provides obstacle detection. The interfacing signals trigger is connected to GP16 and the echo pulse, GP17.



MMBasic provides the DISTANCE function(trigger, echo) for interfacing with the SR04 easily.

Under command mode, we can enter and print the function distance directly. Try placing the object in front of the SR04 module, adjusting the distance, and clicking the UP arrow to repeat the command.

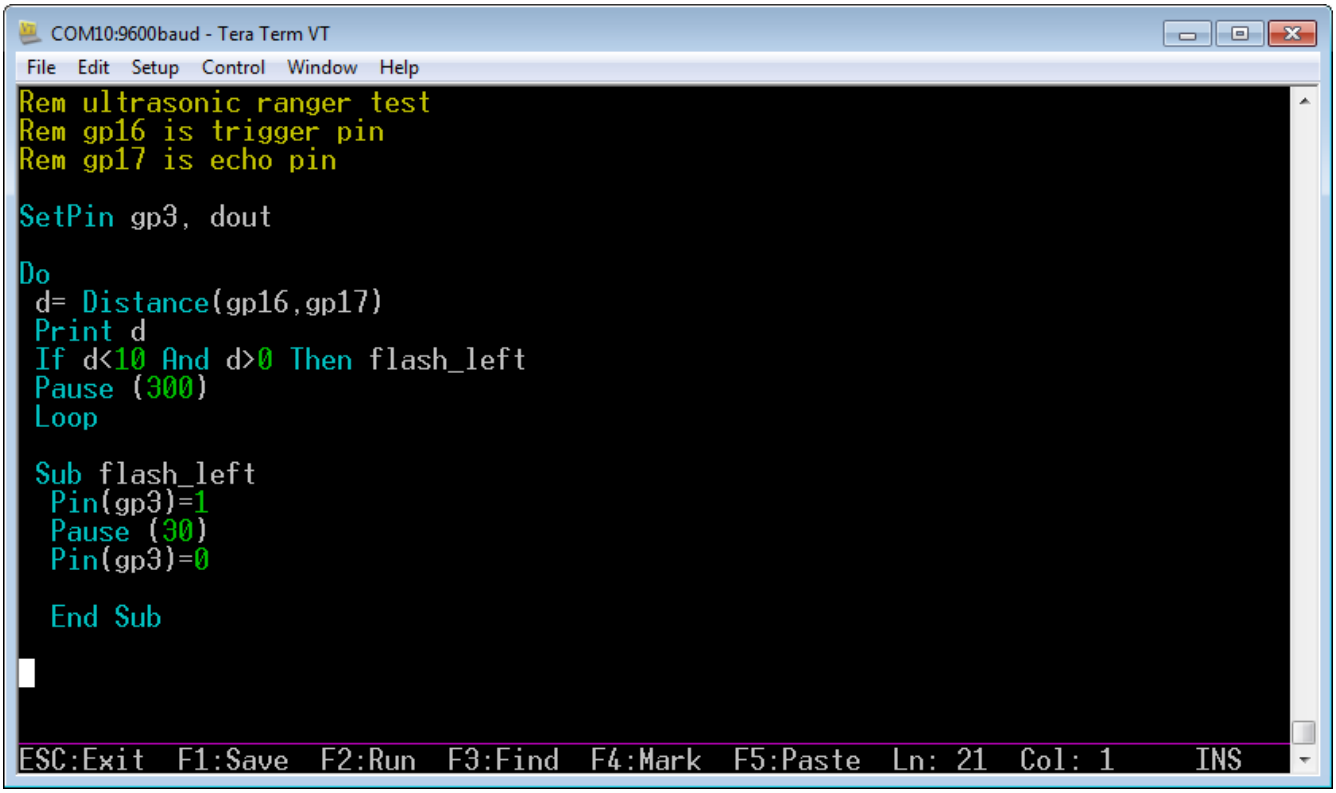
```
>
>
> print distance(gp16,gp17)
 8.293103448
> print distance(gp16,gp17)
 7
> print distance(gp16,gp17)
 6.689655172
> print distance(gp16,gp17)
```

```
 7
> print distance(gp16,gp17)
 9.689655172
>
```

The distance is in centimeter units. Significant digit is about 0.1.

We can write a program that uses distance data for controlling the bot response.

Edit the program below using the built-in editor.



```
Rem ultrasonic ranger test
Rem gp16 is trigger pin
Rem gp17 is echo pin

SetPin gp3, dout

Do
 d= Distance(gp16,gp17)
 Print d
 If d<10 And d>0 Then flash_left
 Pause (300)
 Loop

 Sub flash_left
  Pin(gp3)=1
  Pause (30)
  Pin(gp3)=0

  End Sub
```

The main program is do-loop forever, reads the distance from the ultrasonic module, prints it on screen, and detects the range. If it is less than 10cm and d>0, then make a flash light.

The flash_left is a subroutine that turns on a GP3 LED.

Run the program, place a piece of paper in front of the ultrasonic sensor, move it and see what is happening.

Can you change the detection range? How?

Can you make both headlights flash? How?

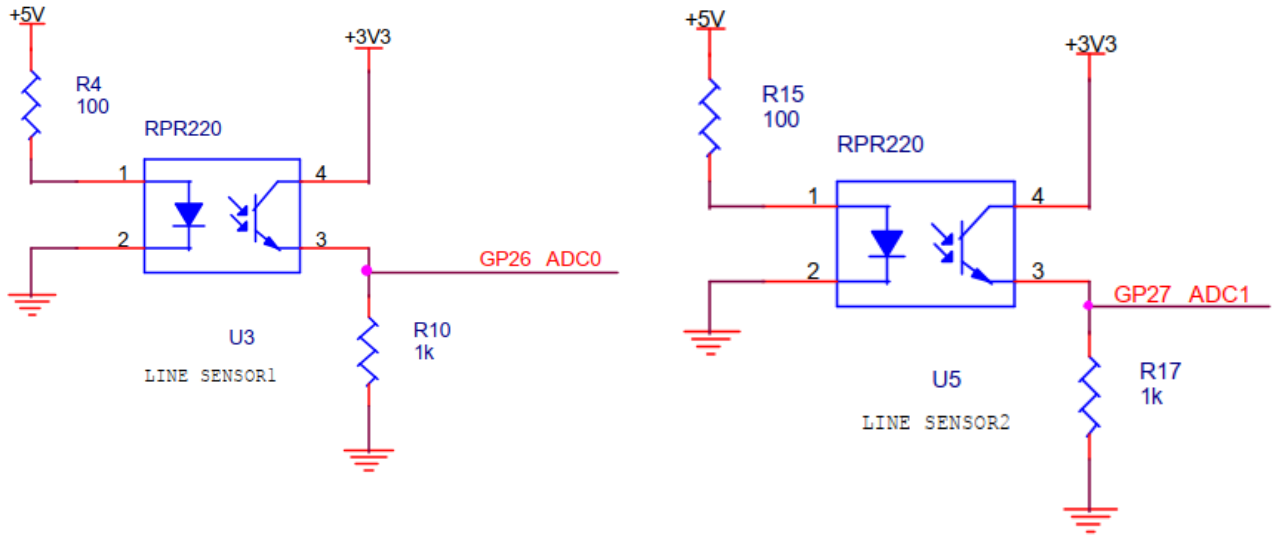
LAB 8 Line sensor

1. Detecting the black line with photo reflective sensor

2. Calibrating the sensors

The black line sensor is made of the IR photo reflective method. The photo reflective sensor, RPR220, has an IR LED and phototransistor. The IR LED, pin 1-2 is forward bias at 100 Ohms, producing an IR wave. The IR wave is transmitted to the floor and reflected back to the phototransistor, pin 3-4. The photogenerated current is detected by R10. The voltage developed on R10 is changed with the amount of reflected IR wave.

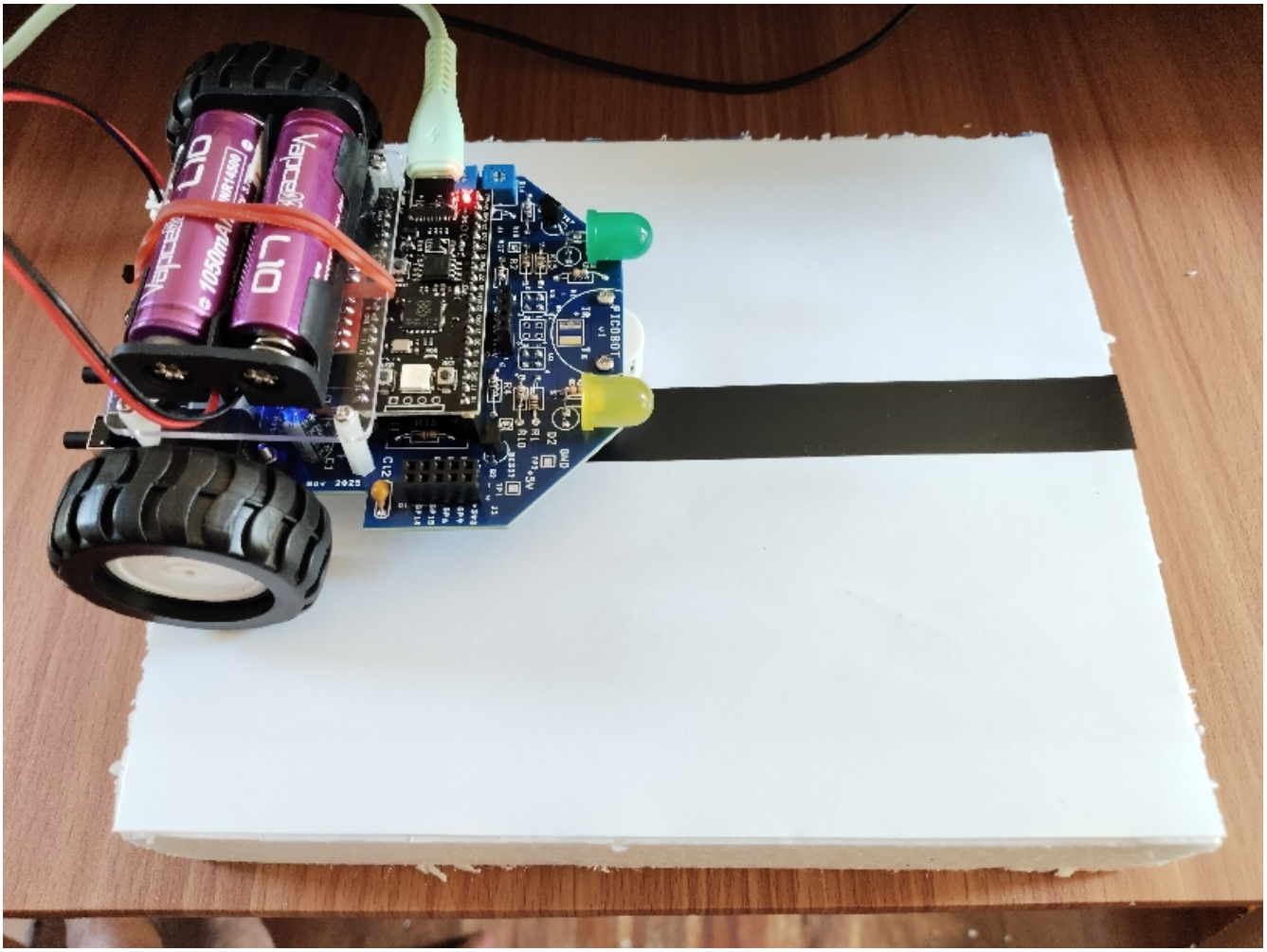
This signal is connected to the analog to digital converter, ADC0. MMBasic has a function pin that reads analog input.

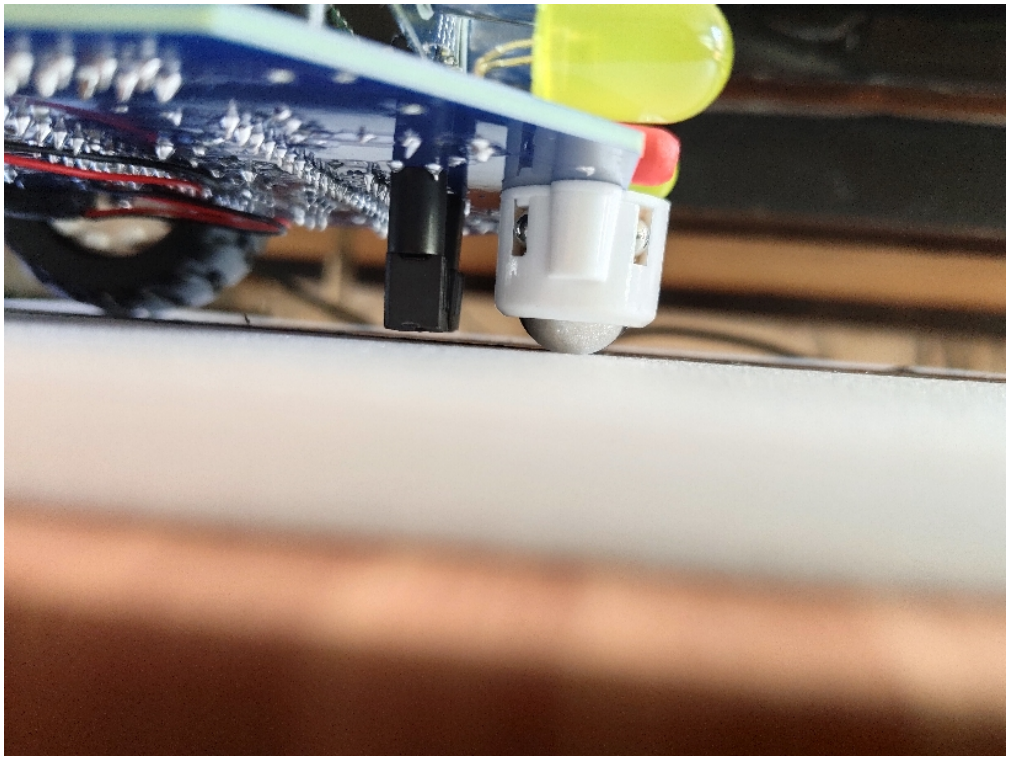


This version has two sensors for black line detection, GP26 and GP27.

We can make a testing floor using white paper pasted on a 16x16 cm form pad, and a black PVC tape.

Place the bot on the white floor and check the readings and on the black tape





The height of the photo sensor from the floor is approx. 5mm

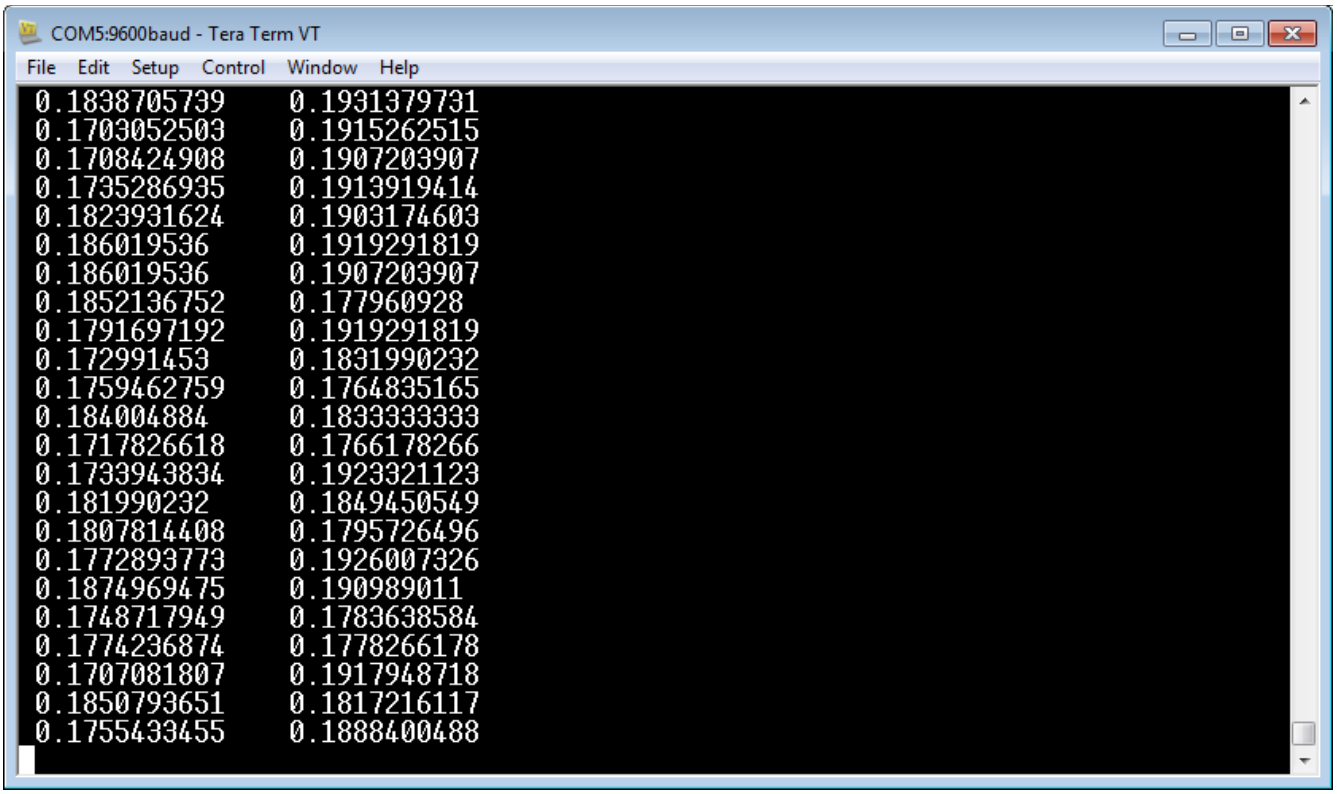
The test code is shown below.

```
Rem photosensor test

SetPin gp26,ain
SetPin gp27,ain

Do
 Print Pin(gp26), : Print Pin(gp27)
 Pause (300)
 Loop
```

The pico board has three analog inputs, GP26, GP27 and GP28.
We can set GP26 and GP27 to be analog input with the command Setpin easily.

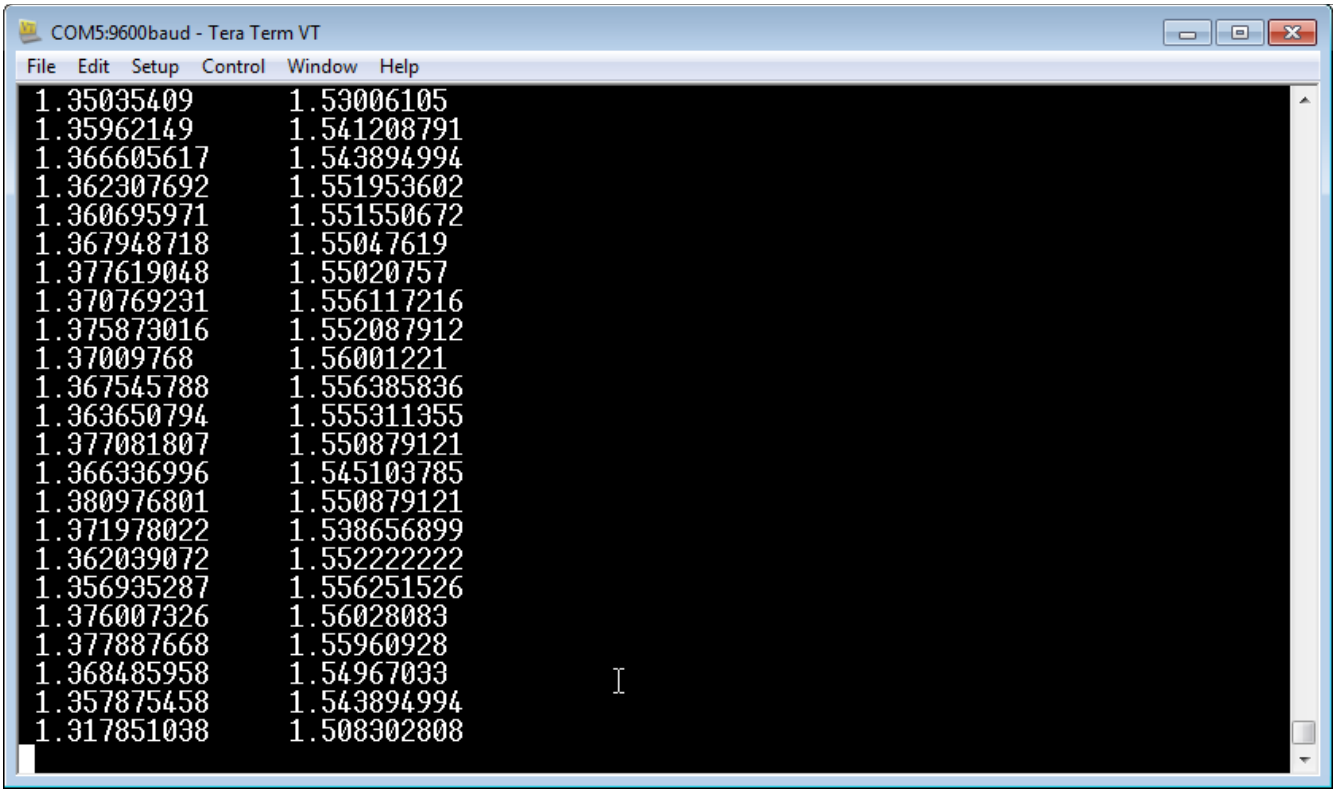
The main program is do-loop forever, printing the analog input ADC0 and ADC1.

Run the program and place the bot on the floor.



The above results show the readings in the Volt from both sensors on the black line.

Now, if we move the bot to the white floor, we will get these results.

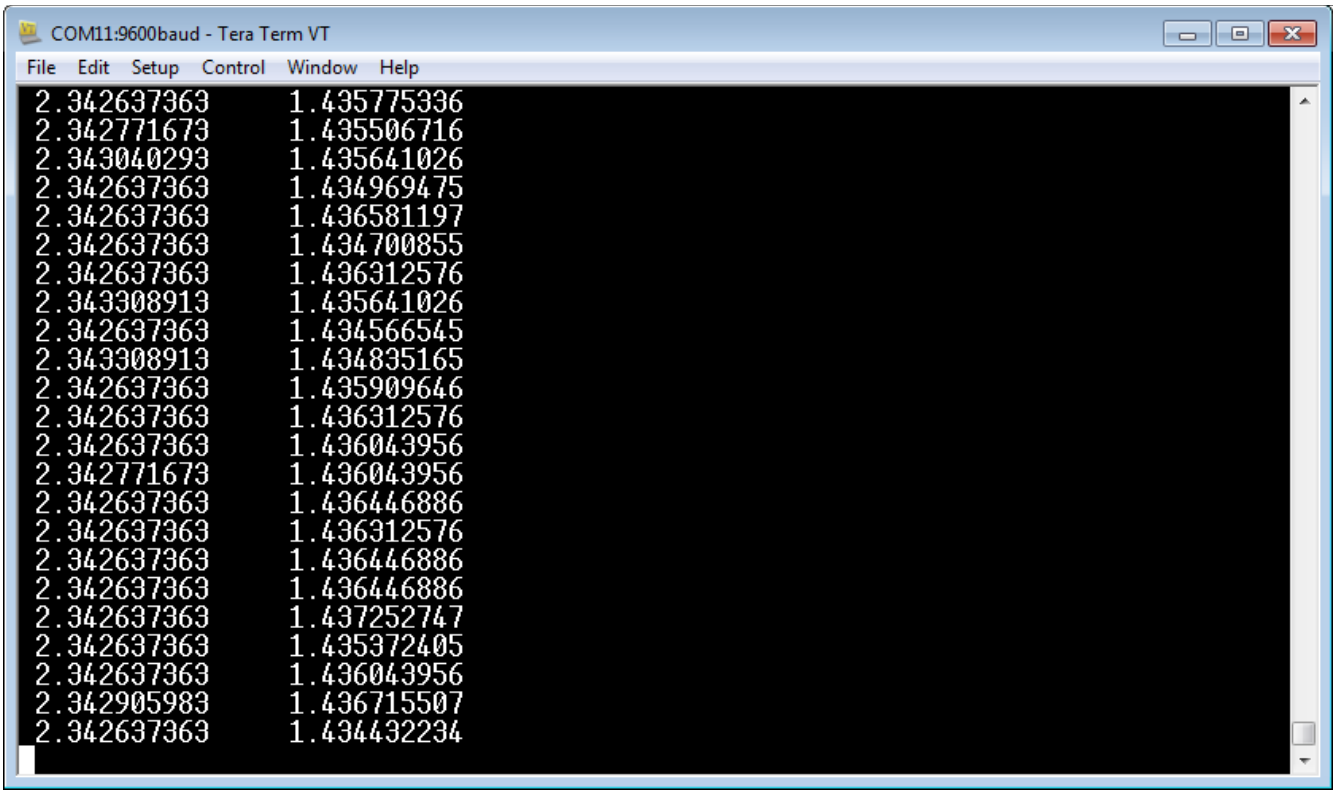


Did you see the difference?

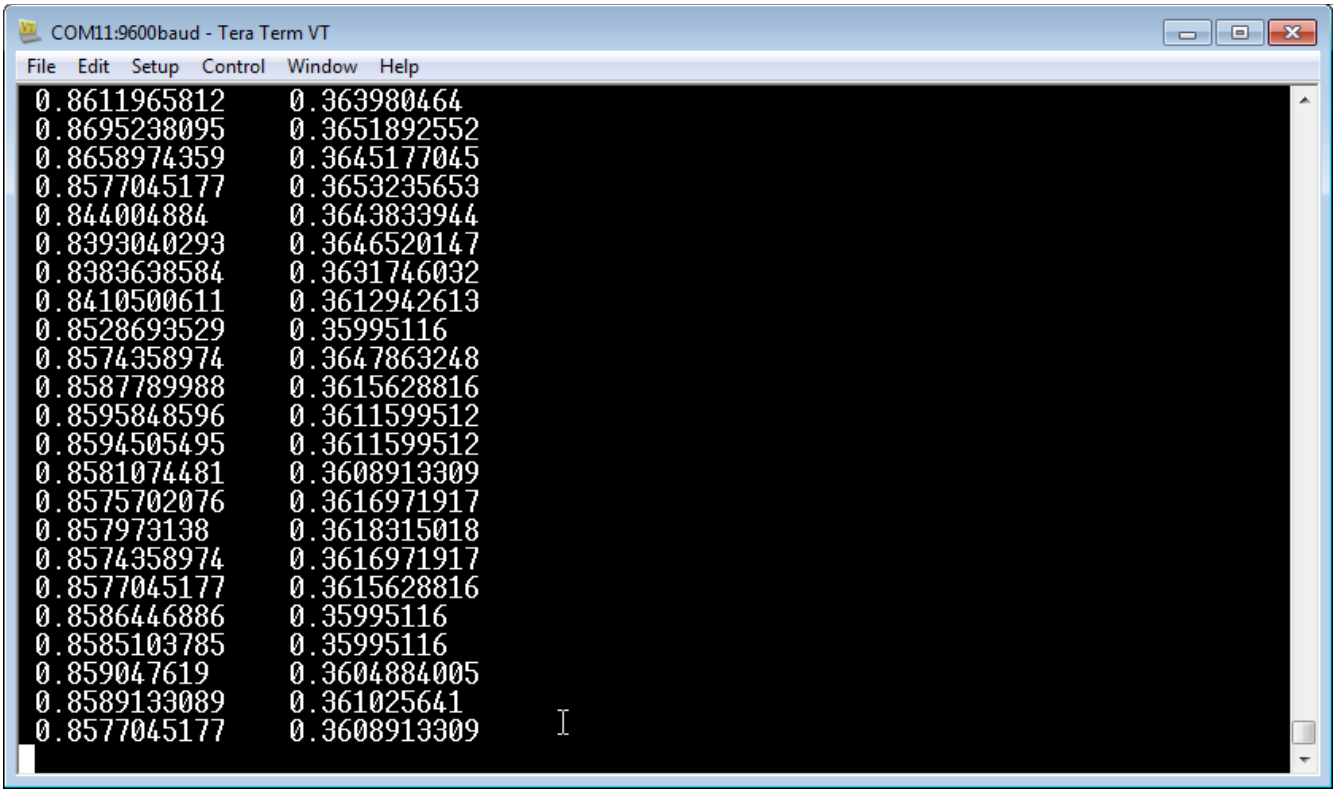
The black tape absorbs much of the IR wave, thus less amount of reflected wave. Photocurrent is then generated less.

We see that even with the same surface, e.g., white colored floor, the readings from both sensors are a bit different.

Here is another bot readings on the white surface.



And on the black tape.



We see that the first sensor gives higher value.

How can we adjust it to produce nearly the same readings?

```
Rem photosensor calibration

SetPin gp26,ain
SetPin gp27,ain

SetPin gp2,dout
SetPin gp3,dout

gain=1

Do

 A = Pin(gp26)*gain : B = Pin(gp27)

 Print A, : Print B
```

```
Pause (300)

Loop
```

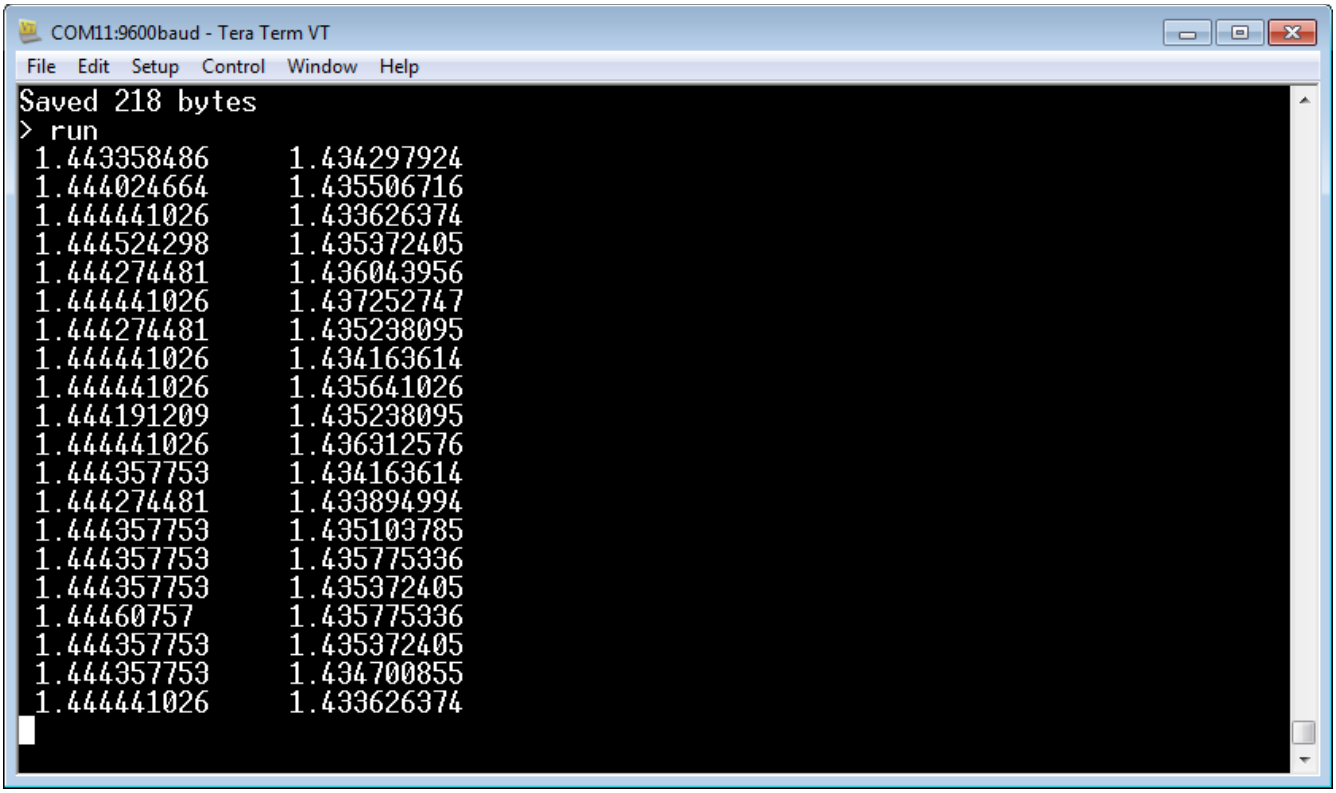
Run this program again.

If we change the gain from 1 to 0.62.

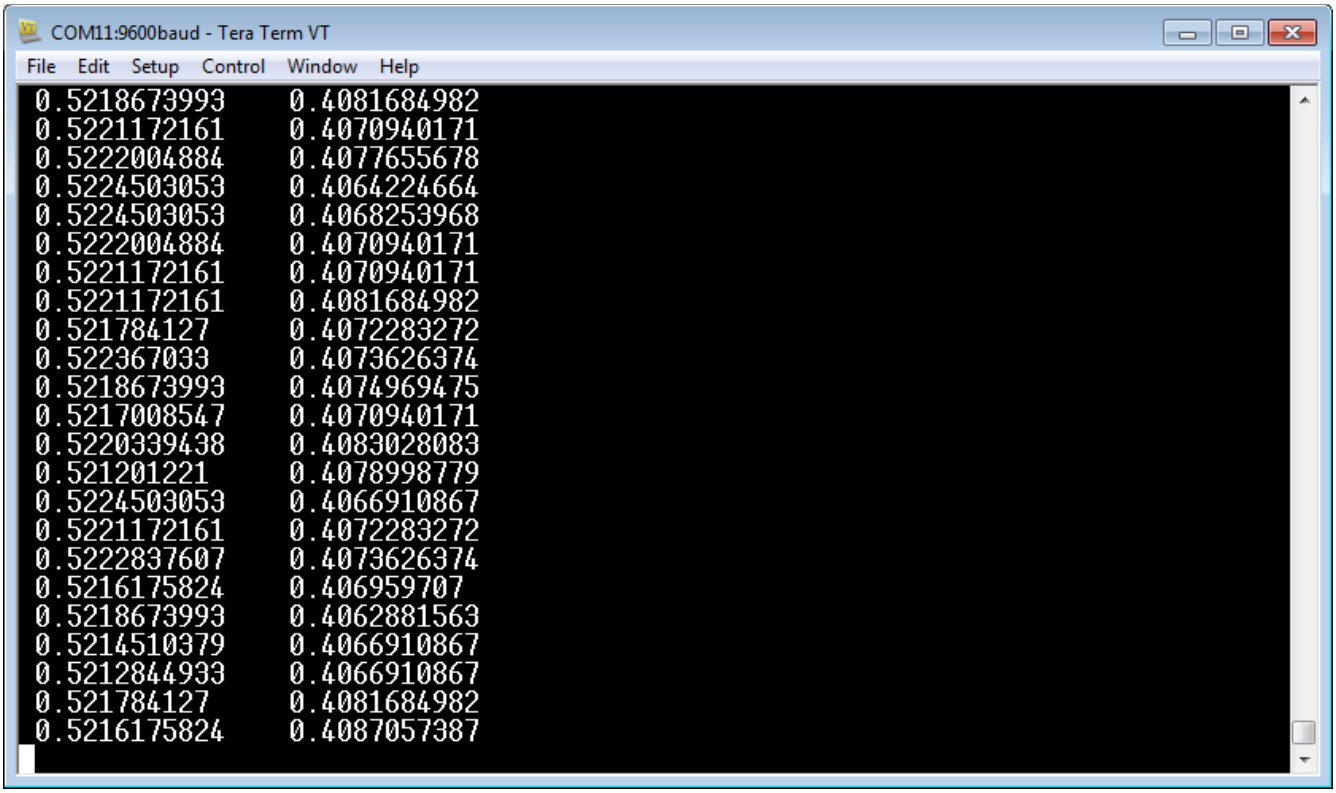
```
Rem photosensor calibration

SetPin gp26,ain
SetPin gp27,ain

SetPin gp2,dout
SetPin gp3,dout

gain=0.62

Do
 A = Pin(gp26)*gain : B = Pin(gp27)
 Print A, : Print B
 Pause (300)
 Loop
```

Run above program with new gain, 0.62.

Place the bot on white floor.



And then on black tape.



We see that simple calibration with such gain adjustment gives better readings.

We will use these sensor readings for line tracking in a later lab.

For now, if we want to have a status flag using headlights, turn it on when the bot detects a black line. How to do that?

```
Rem photosensor calibration

SetPin gp26,ain
SetPin gp27,ain

SetPin gp2,dout
SetPin gp3,dout

gain=0.62

Do

 A = Pin(gp26)*gain : B = Pin(gp27)

 Print A, : Print B

 If A>1.2 Then Pin(gp3)=0
 If A<0.8 Then Pin(gp3)=1

 If B>1.2 Then Pin(gp2)=0
 If B<0.8 Then Pin(gp2)=1


 Pause (300)

 Loop
```

We can add If Then statement for detecting the readings of white floor and black line.

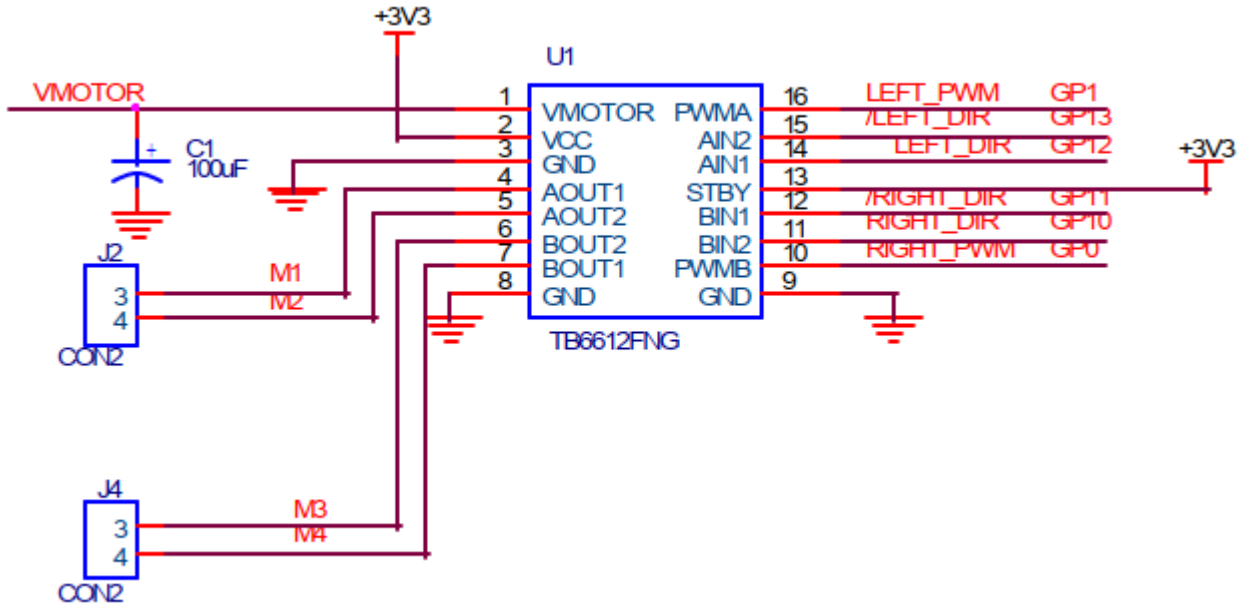
Now if we place the bot sensor on a black line, did you get the headlights ON?

LAB 9 Motor control 1 : H-bridge and PWM

1. Control the motor with an H-bridge driver

2. Speed control using PWM

The bot has two wheels driven by an N20 gear motor. The power for each motor is supplied by a TB6612 module, a 2-channel H-bridge motor driver. The module can change motor spinning directly, CW or CCW and change the power using a PWM signal. The controller under the BASIC program will provide logic control for speed and direction.



GP0 and GP1 generate a PWM signal for power controlling the left and right motors. The direction of the left motor is controlled by signal /LEFT_DIR (GP13) and LEFT_DIR (GP12).

/LEFT_DIR, GP13	LEFT_DIR, GP12	Spinning
0	1	CW
1	0	CCW
1	1	Stop
0	0	Stop

And the power supplied to the motor, VMOTOR +8V, is then shipped by signal LEFT_PWM, GP1. The PWM frequency is 500Hz, and the duty cycle can be varied from 0 to 100%.

Under command prompt, try the below settings. (for a motor control experiment, we need a battery and switch ON the battery power.)

You may need the bot on a stand, to make the wheel free to spin.

```
> setpin gp12,dout
> setpin gp13,dout
>
> setpin gp0,pwm0A
> setpin gp1,pwm0B
>
> pin(gp12)=1
> pin(gp13)=0
> pwm 0,500,50,50
>
```

Second, we set GP0 and GP1 to be PWM channel 0A and channel 0B.

Third, we set gp12 and gp13, 1 and 0. And set PWM channel 0 to 500Hz, 50% for both channels.

We will see motorists spinning.

To stop the motor, we can set duty cycle to 0%.

>PWM 0,500,0,0

Can you change spinning direction? How?

Can you change motor spinning speed? How?

If we want to control the spinning speed, from 0% to 100%, we can try the below command.

```
>
```

```
> for i=0 to 100 : pwm 0,500,i,i : pause(100) : next
>
```

Now the duty cycle for both channels will be variable I. For the next loop it will increment I buy one every 100ms.
If we want to slow down from 100% to 0%, we can try below command.

```
>
> for i=100 to 0 step -1 : pwm 0,500,i,i : pause(100) : next
>
```

Each step, the variable I will be decremented by 1 every 100ms.

For the right motor, the control logic will be as follows.

/RIGHT_DIR, GP11	RIGHT_DIR, GP10	Spinning
0	1	CW
1	0	CCW
1	1	Stop
0	0	Stop

Can you control the right motor with similar commands? How?

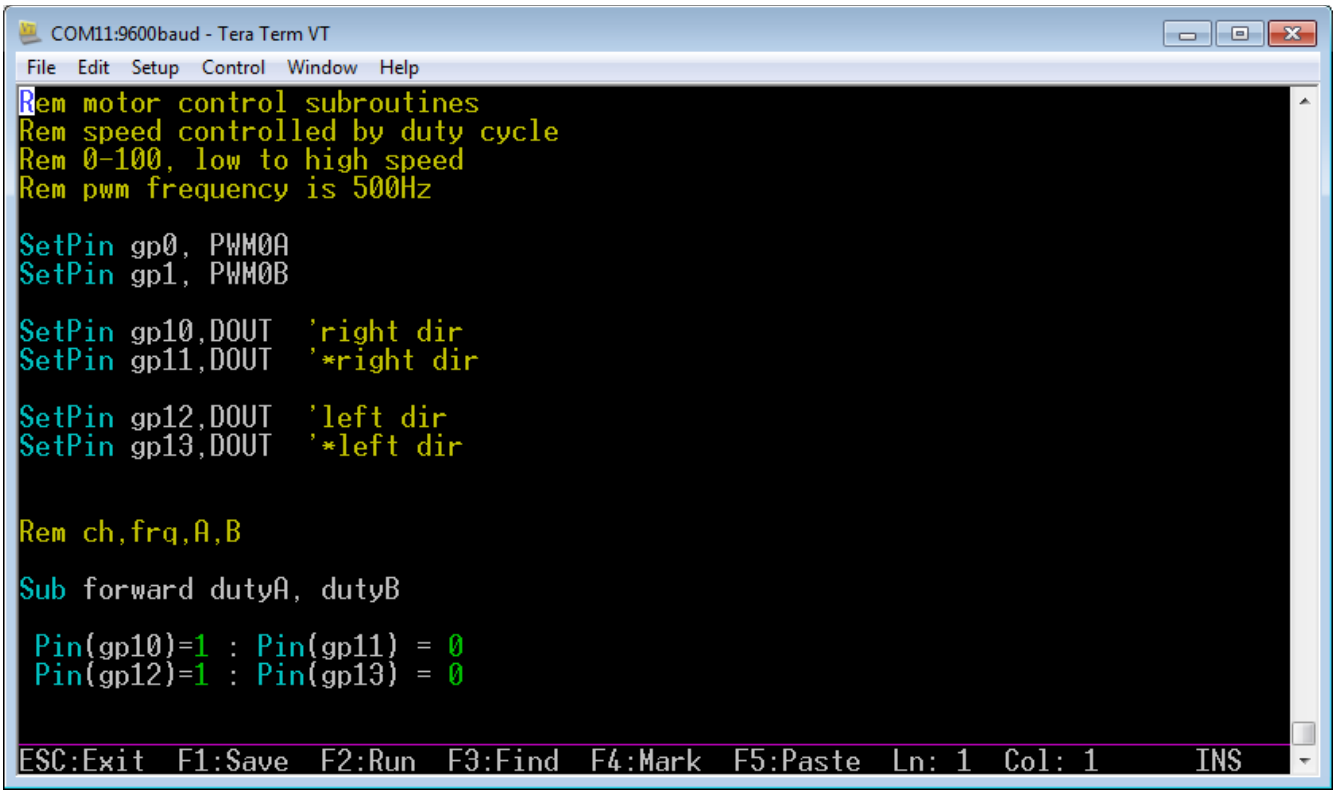
LAB 10 Motor control 2 : basic movements

1. Forward, reverse movement

2. Turn left and turn right

From LAB 8 we learned how to the motor direction using h-bridge and speed adjust using PWM. Now we will set up subroutines that control the bot movement, forward, reverse, turn left and turn right.

To write the subroutine, we use Editor, under prompt>Edit.



To test the program, we will put the bot on stand. And see the wheel movement.

Here is the list of subroutine forward.

```
SetPin gp0, PWM0A
SetPin gp1, PWM0B

SetPin gp10,DOUT  'right dir
SetPin gp11,DOUT  '*right dir

SetPin gp12,DOUT  'left dir
SetPin gp13,DOUT  '*left dir


Rem ch,frq,A,B

Sub forward dutyA, dutyB

 Pin(gp10)=1 : Pin(gp11) = 0
 Pin(gp12)=1 : Pin(gp13) = 0

PWM 0,500,dutyA, dutyB

  End Sub
```

All ports are set up for pin functions. Subroutine forward will get two parameters, duty cycle for PWM channel A and B.

To make both motors spin forward. The H-bridge control bits are set as shown above. Then the function PWM channel 0 will generate a PWM signal at 500Hz with duty cycle for channel A and channel B.

When you completely edit the program, press F1, to save and exit. Then run the program.

Under prompt mode, we can test motor running with a new command, forward dutyA, dutyB directly.

```
> run
> forward 10,10
> forward 0,0
```

```
> forward 100,100
> forward 40,50
> forward 50,50
>
> forward 0,0
```

To stop motor running, type command> forward 0,0

Again, we can test motor running from 0 to 50% power with below commands.

```
>for i=0 to 50 : forward i,i : pause(100) : next
```

Similarly, we can test forward speed from 100% to 0%.

```
>
> for i=100 to 0 step -1 : forward i,i : pause(100) : next
>
```

So, we have experimented with forward direction.

How to make reverse direction? Or turn left, turn right.

```
Sub reverse dutyA, dutyB

 Pin(gp10)=0 : Pin(gp11) = 1
 Pin(gp12)=0 : Pin(gp13) = 1

PWM 0,500,dutyA, dutyB

  End Sub
```

The subroutine reverse is similar to forward. Only the opposite control at the h-bridge circuit.

We can add this subroutine after forward subroutine. When complete editing, run the program. And test the reverse command.

```
> run
> reverse 10,10
>
>
```

To turn the bot left or right, we can make the motor left and right spinning in opposite direction.

```
Sub turnright dutyA, dutyB

 Pin(gp10)=0 : Pin(gp11) = 1
 Pin(gp12)=1 : Pin(gp13) = 0

PWM 0,500,dutyA, dutyB

  End Sub


Sub turnleft dutyA, dutyB

 Pin(gp10)=1 : Pin(gp11) = 0
 Pin(gp12)=0 : Pin(gp13) = 1

PWM 0,500,dutyA, dutyB

  End Sub
```

What about stop command for stop motor running? We can add it also.

```
Sub stop

 Pin(gp10)=0 : Pin(gp11) = 0
 Pin(gp12)=0 : Pin(gp13) = 0

  End Sub
```

Now we can test motor running with these new commands

Forward DutyA, DutyB	Forward 50,50

Reverse DutyA, DutyB	Reverse 10,10
Turnleft DutyA, DutyB	Turnleft 30,30
Turnright DutyA, DutyB	Turnright 30,30
Stop	Stop

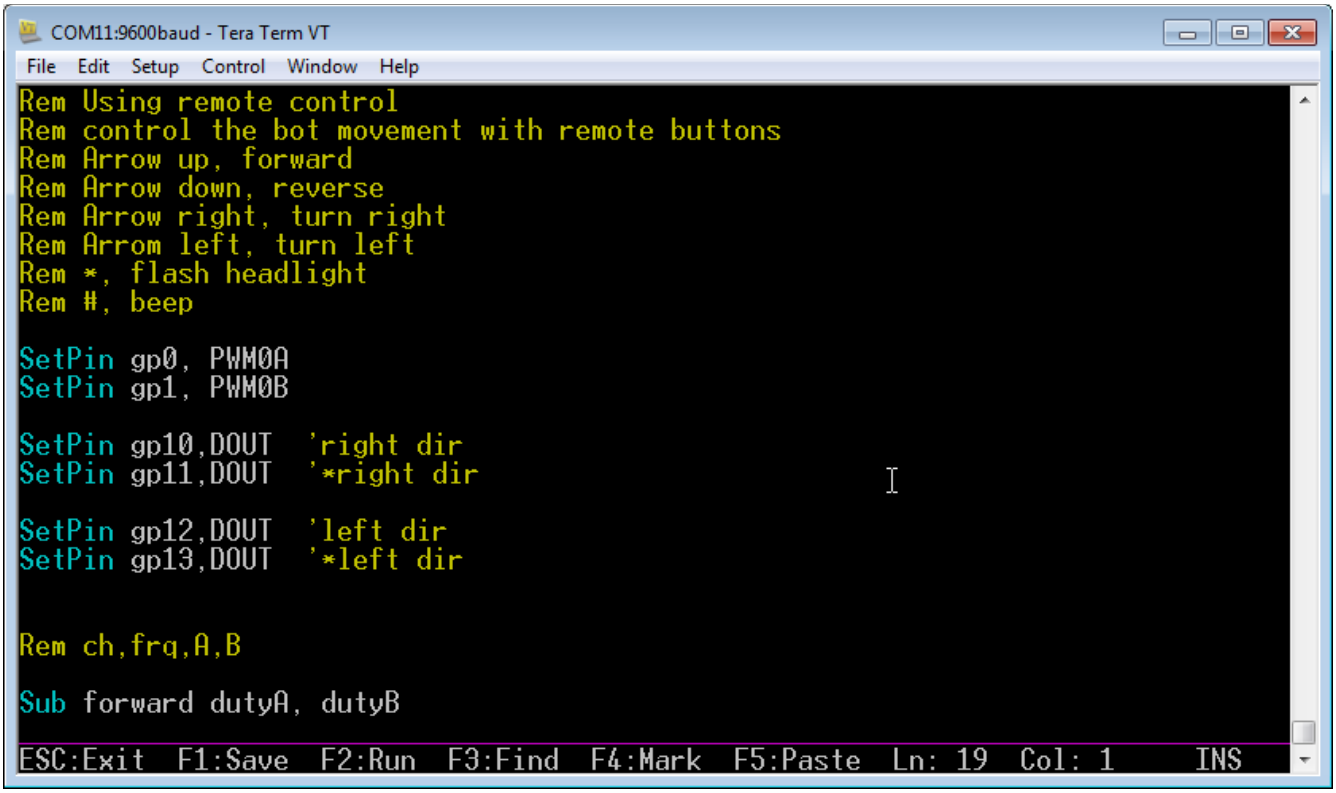
We can test motor running and adjust speed under command prompt directly.

LAB 11 Control the bot using remote control

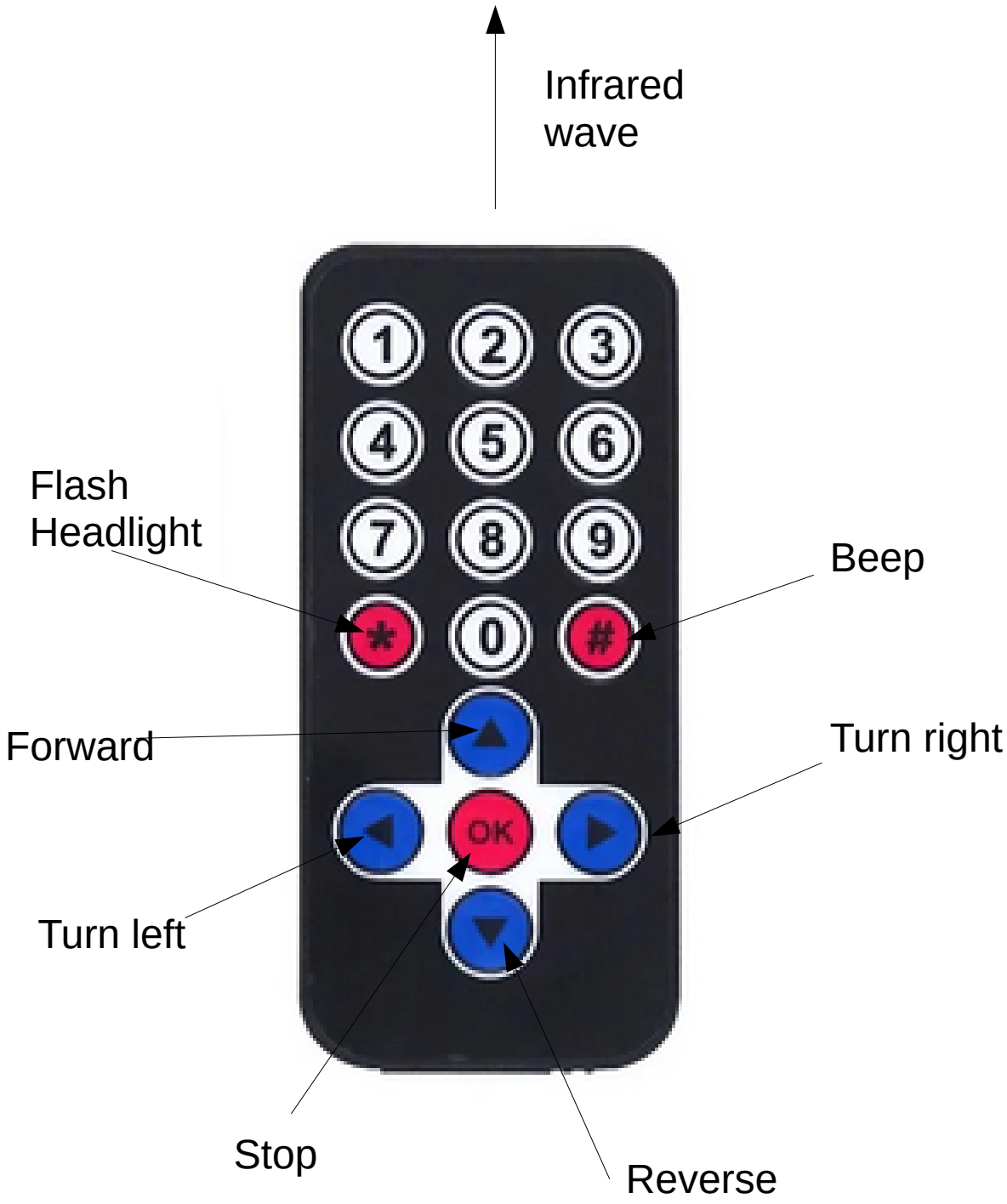
1. Test the bot moving with remote control

2. Learn Select-case statement

We will use the remote control to control the bot manually. This program is quite long, we will use an editor to edit the program. Under command prompt>Edit.



We will point the remote control to the IR receiver on the rear side of the bot.



Here is the list of remote control program.

```
Rem Using remote control
Rem control the bot movement with remote buttons
Rem Arrow up, forward
Rem Arrow down, reverse
Rem Arrow right, turn right
Rem Arrom left, turn left
Rem *, flash headlight
Rem #, beep

SetPin gp0, PWM0A
SetPin gp1, PWM0B

SetPin gp10,DOUT  'right dir
SetPin gp11,DOUT  '*right dir

SetPin gp12,DOUT  'left dir
SetPin gp13,DOUT  '*left dir


Rem ch,frq,A,B

Sub forward dutyA, dutyB

 Pin(gp10)=1 : Pin(gp11) = 0
 Pin(gp12)=1 : Pin(gp13) = 0

PWM 0,500,dutyA, dutyB

  End Sub

Sub reverse dutyA, dutyB

 Pin(gp10)=0 : Pin(gp11) = 1
 Pin(gp12)=0 : Pin(gp13) = 1

PWM 0,500,dutyA, dutyB

  End Sub
```

```
Sub stop

 Pin(gp10)=1 : Pin(gp11) = 1
 Pin(gp12)=1 : Pin(gp13) = 1

  End Sub


Sub turnright dutyA, dutyB

 Pin(gp10)=0 : Pin(gp11) = 1
 Pin(gp12)=1 : Pin(gp13) = 0

PWM 0,500,dutyA, dutyB

  End Sub


Sub turnleft dutyA, dutyB

 Pin(gp10)=1 : Pin(gp11) = 0
 Pin(gp12)=0 : Pin(gp13) = 1

PWM 0,500,dutyA, dutyB

  End Sub

SetPin gp2, dout
SetPin gp3, dout

Sub flash_light
  Pin(gp2)=1 : Pin(gp3)=1 : Pause (30)
  Pin(gp2)=0 : Pin(gp3)=0 : Pause (1000)

  End Sub
```

```
SetPin gp4, ir
Dim integer devcode, keycode
IR devcode, keycode, irint

Do
‘ main body loop
 Loop

 Sub irint
  Print devcode, keycode

  Select Case keycode

  Case 56 : stop
  Case 24 : forward 50,50
  Case 74 : reverse 30,30
  Case 90 : turnright 30,30
  Case 16 : turnleft 30,30
  Case 176 : Play tone 500,415,200 : Pause (500)
  Case 104 : flash_light

  End Select

  End Sub
```

The main program is do-loop forever. The IR code is an interrupt subroutine, IRINT. When it receives the IR wave, MMBasic will decode and the select case will check the key code and run the functions. The IR remote control code we have learned from LAB 5.

Testing the bot running, will need battery power.

Let us try to place the bot on a stand. Run the program, then use the remote control.

Can you see the motor running, flashing lights and beep when a given remote key is pressed?

If we point the remote control to the room wall, does it work? Why?

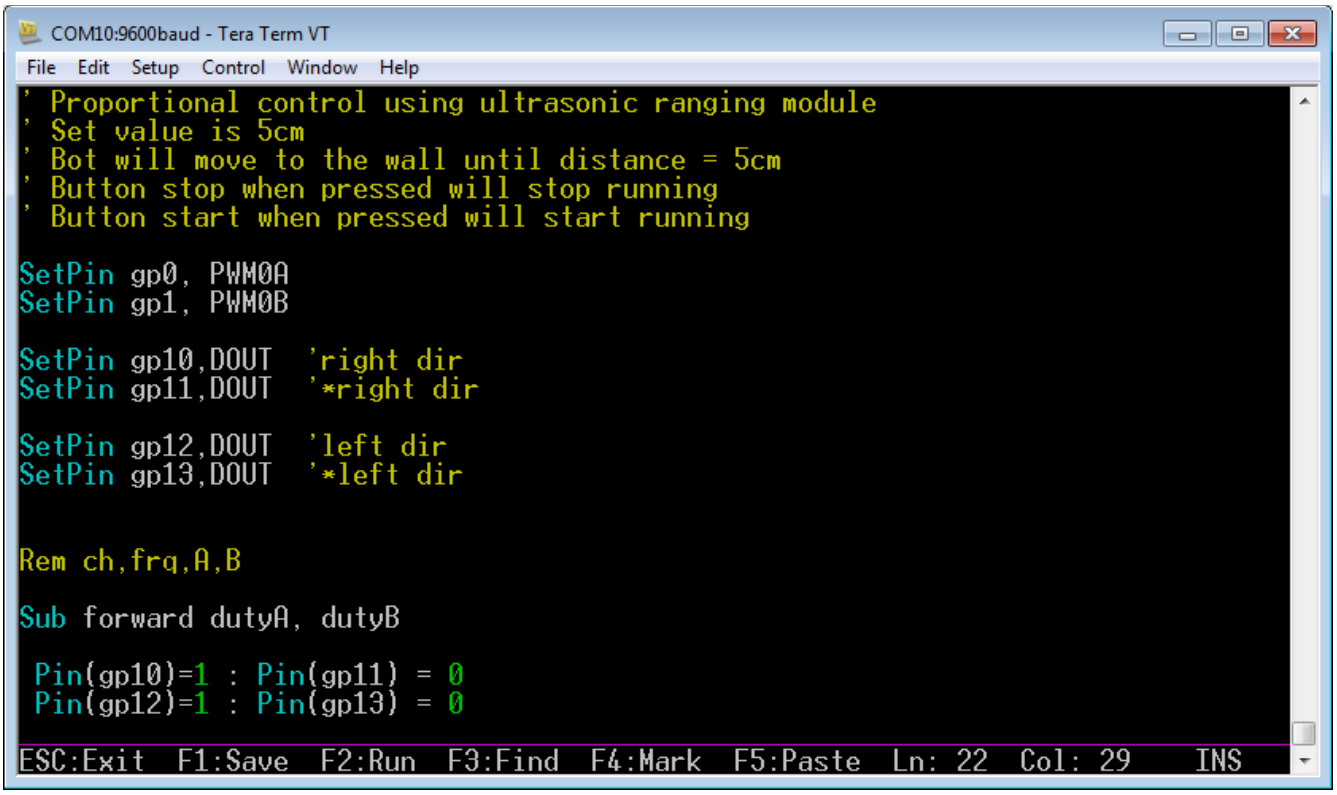
Can you make other keys for controlling bots? How?

LAB 12 Proportional control using ultrasonic ranger

1. Test the bot moving with an ultrasonic ranger

2. Learn the basics of proportional control

We will use the ultrasonic ranger module as the bot sensor. The bot will move toward the wall and when it gets close to the wall, the speed will slow down until stop. The control method is proportional control. This program is quite long, we will use an editor to edit the program.



Here is the list of LAB 10 program.

```
' Proportional control using ultrasonic ranging module
' Set value is 5cm
' Bot will move to the wall until distance = 5cm
' Button stop when pressed will stop running
' Button start when pressed will start running

SetPin gp0, PWM0A
SetPin gp1, PWM0B

SetPin gp10,DOUT  'right dir
SetPin gp11,DOUT  '*right dir

SetPin gp12,DOUT  'left dir
SetPin gp13,DOUT  '*left dir


Rem ch,frq,A,B

Sub forward dutyA, dutyB

 Pin(gp10)=1 : Pin(gp11) = 0
 Pin(gp12)=1 : Pin(gp13) = 0

PWM 0,500,dutyA, dutyB

  End Sub

Sub reverse dutyA, dutyB

 Pin(gp10)=0 : Pin(gp11) = 1
 Pin(gp12)=0 : Pin(gp13) = 1

PWM 0,500,dutyA, dutyB

  End Sub

Sub stop
```

```
 Pin(gp10)=1 : Pin(gp11) = 1
 Pin(gp12)=1 : Pin(gp13) = 1

  End Sub


Sub turnright dutyA, dutyB

 Pin(gp10)=0 : Pin(gp11) = 1
 Pin(gp12)=1 : Pin(gp13) = 0

PWM 0,500,dutyA, dutyB

  End Sub


Sub turnleft dutyA, dutyB

 Pin(gp10)=1 : Pin(gp11) = 0
 Pin(gp12)=0 : Pin(gp13) = 1

PWM 0,500,dutyA, dutyB


  End Sub

Sub flash_left
  Pin(gp3)=1
  Pause (30)
  Pin(gp3)=0


  End Sub
```

```
' start main loop

SetPin gp7, din, pullup
SetPin gp8, din, pullup

SetPin gp3, dout

Kp = 1.5

start:

Do While Pin(gp8)=1

 d= Distance(gp16,gp17)

 err = d-5     'set value = 5 cm

 If err<0 Then err=0

 Pout = Int(err*Kp)

 If Pout>100 Then Pout=100

  forward(Pout,Pout)

 Print d,Pout

 If d<=10 And d>0 Then flash_left
 Pause (300)
Loop
 stop
 Pause (500)

Do While Pin(gp7)=1
 Pause (100)
  Loop
  GoTo start
```

The main program reads the distance from the ultrasonic ranger, computes the error value with a set point of 5cm. The power output is computed with Kp*err. We can change the gain, Kp to see the response. The stop button is used to stop the motor and the start button will restart the bot.

To test the bot moving, we will set the option AUTORUN ON with a prompt command.

>OPTION AUTORON ON

Check the current option using >option list

```
> option list
PicoMite MMBasic RP2040 Edition V6.00.03
OPTION AUTORUN  ON
OPTION COLOURCODE ON
OPTION CPUSPEED (KHz) 200000
OPTION AUDIO GP18,GP19', ON PWM CHANNEL 1
>
```

The option AUTORUN ON, will make the bot run the program without RUN at the command prompt.

We can place the bot on the floor and have a wall for ultrasonic detection.

When power is on, the bot will move to the wall and will stop at approx. 5cm.

Can you change the set value from 5cm to 10cm? How?

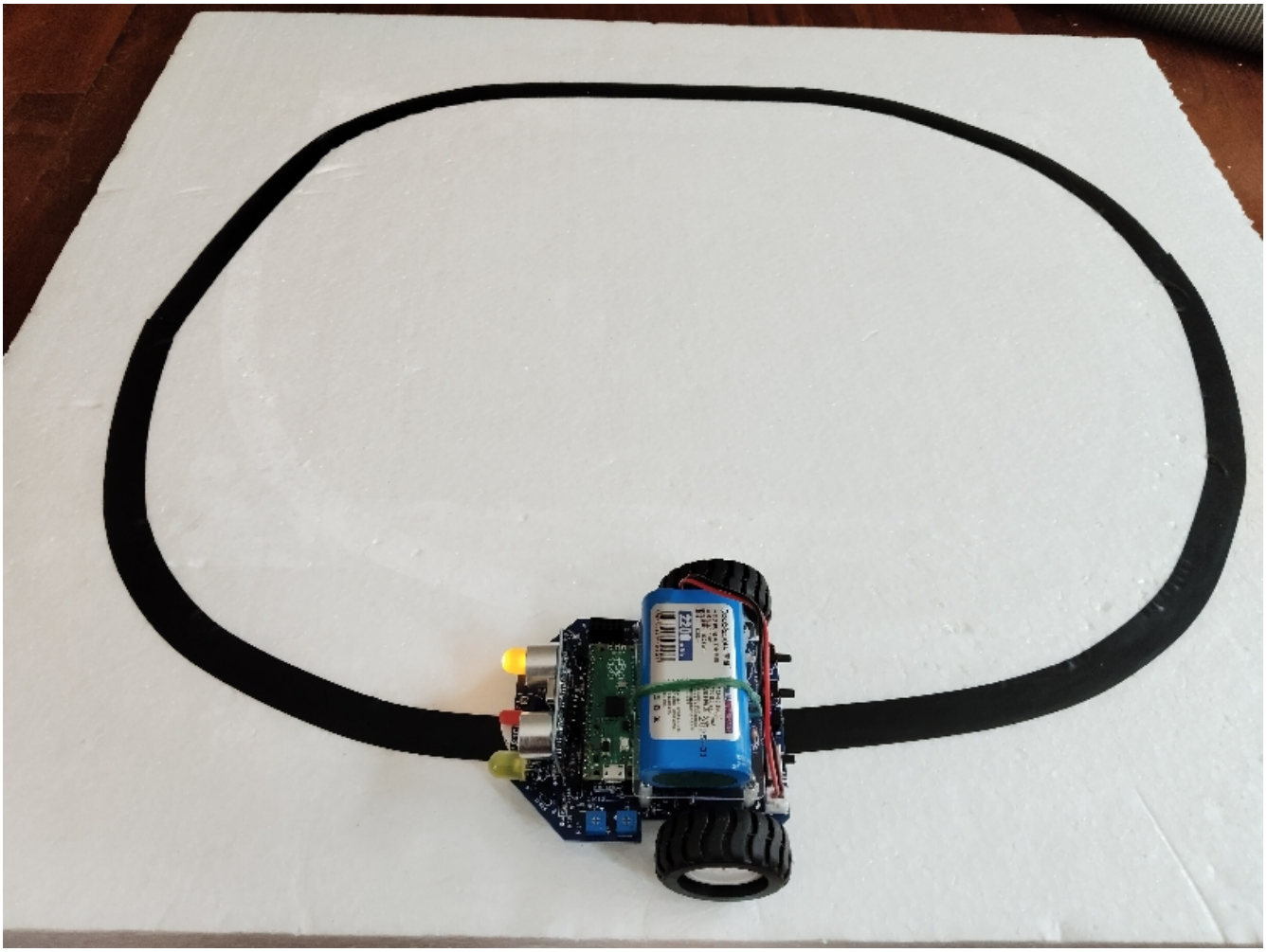
If we change the proportional gain, Kp, what will happen?

LAB 13 Line tracking with PD control

1. Using proportional-derivative control

2. Test the bot with black line tracking

We will experiment with the bot for line tracking. The control method is proportional-derivative, PD control. The bot uses two photo reflective sensors connected to ADC0 and ADC1 pins. The control methods are proportional-derivative, PD control. Errors will be computed from both sensors.



Here is the program listing.

```
'Line tracking
'computes error, adjust steering

SetPin gp0, PWM0A
SetPin gp1, PWM0B

SetPin gp10,DOUT  'right dir
SetPin gp11,DOUT  '*right dir

SetPin gp12,DOUT  'left dir
SetPin gp13,DOUT  '*left dir


Rem ch,frq,A,B

Sub forward dutyA, dutyB

 Pin(gp10)=1 : Pin(gp11) = 0
 Pin(gp12)=1 : Pin(gp13) = 0

PWM 0,500,dutyA, dutyB

  End Sub

Sub reverse dutyA, dutyB

 Pin(gp10)=0 : Pin(gp11) = 1
 Pin(gp12)=0 : Pin(gp13) = 1

PWM 0,500,dutyA, dutyB

  End Sub

Sub stop
```

```
 Pin(gp10)=0 : Pin(gp11) = 0
 Pin(gp12)=0 : Pin(gp13) = 0

  End Sub


Sub turnright dutyA, dutyB

 Pin(gp10)=0 : Pin(gp11) = 1
 Pin(gp12)=1 : Pin(gp13) = 0

PWM 0,500,dutyA, dutyB

  End Sub


Sub turnleft dutyA, dutyB

 Pin(gp10)=1 : Pin(gp11) = 0
 Pin(gp12)=0 : Pin(gp13) = 1

PWM 0,500,dutyA, dutyB

  End Sub

Sub line_status

 If Pin(gp26)>1.0 Then Pin(gp3) = 0  ' white floor
 If Pin(gp26)<0.8 Then Pin(gp3) = 1  ' black line detected

 If Pin(gp27)>1.0 Then Pin(gp2) = 0  ' white floor
 If Pin(gp27)<0.8 Then Pin(gp2) = 1  ' black line detected

 End Sub

Rem line tracking test

SetPin gp26,ain
SetPin gp27,ain
```

```
SetPin gp2,dout
SetPin gp3,dout

SetPin gp8,din,pullup
SetPin gp7,din,pullup

Kp = 1.5
Kd = 1.2

currentSpeed=30   ‘ set current speed 30

start:

Do While Pin(gp8)=1    ‘ exit if Stop button pressed

 line_status

 err = (Pin(gp26)-Pin(gp27))*3

 P = err
 D = err - last_err

 PD = P*Kp + D*Kd

 Pout = PD

 last_err = err

 Pleft = currentSpeed + Pout
 Pright = currentSpeed - Pout

  Print err,Pleft,Pright

  forward Int(Pleft),Int(Pright)

  Loop

 Stop

  Do While Pin(gp7)=1  ‘ wait start button
```

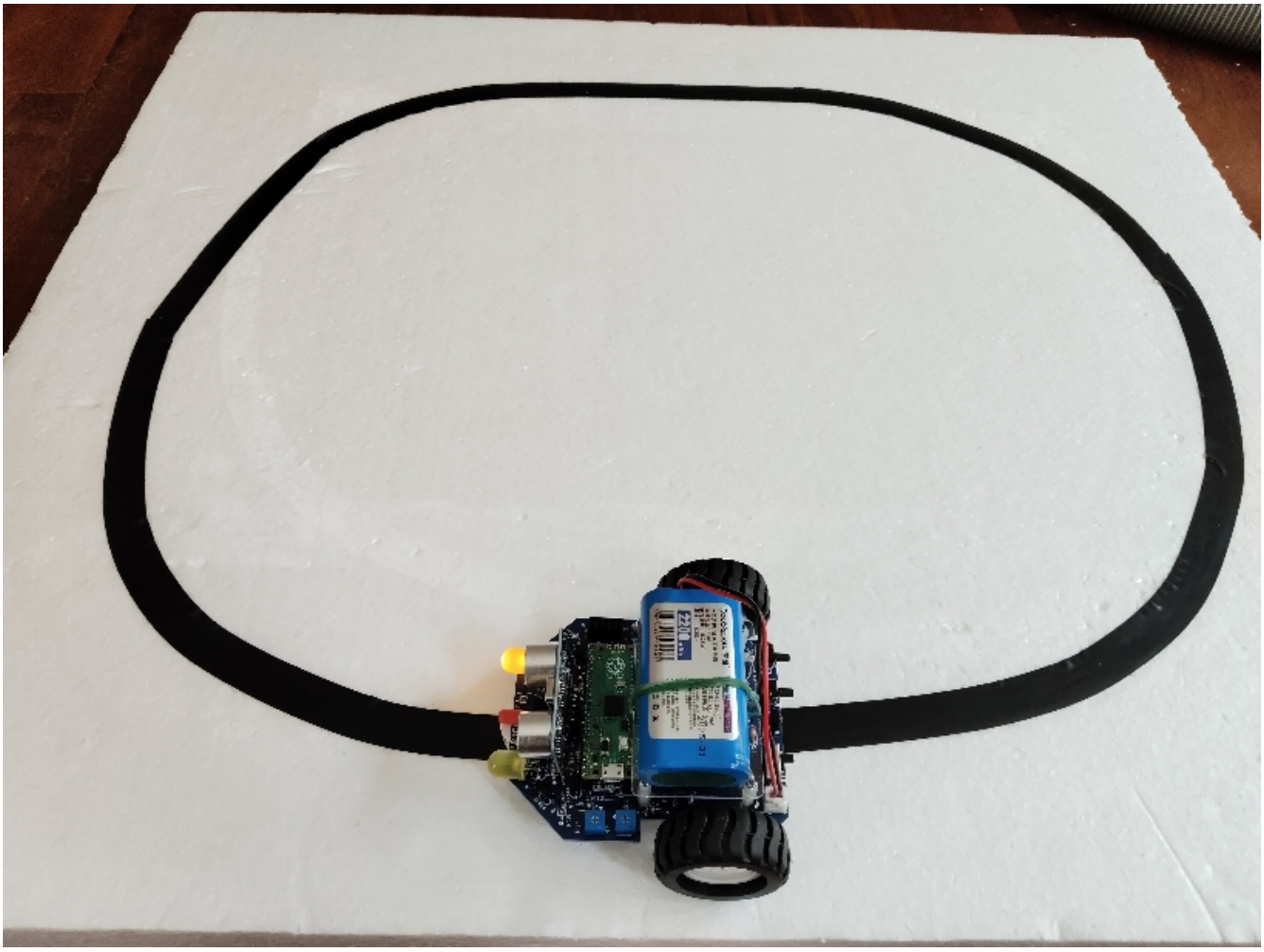
```
Loop
 Pause (2000)
  GoTo start
```

The main program is a do-loop forever reads the voltage from line sensors at ADC0, GP26 and ADC1, GP27, computes the error.

When the sensor detects a black line, both sensors will give nearly zero voltage (results from LAB 7). The error will be changed if the bot moves away from the black line.

This error value will be used to compute the power output for the left and right motors. Speed adjustment for the left and right motors will be regulated using proportional and derivative terms until the error becomes zero. The bot will then track the black line.

The test floor can be made from a foam pad, 60x70cm, and the PVC black tape, 19mm.



Edit the program using Editor. To run the bot on the test floor, set the option AUTORUN ON.

Check the current option with >Option list

```
> option list
PicoMite MMBasic RP2040 Edition V6.00.03
OPTION AUTORUN  ON
OPTION COLOURCODE ON
OPTION CPUSPEED (KHz) 200000
OPTION AUDIO GP18,GP19', ON PWM CHANNEL 1
>
```

Place the bot at the starting point. Turn the power switch ON.

The bot will start running on the track.

Did you get the bot running smoothly? Or off the track?

Press STOP button if you you need to stop the motor.

To restart, press the START button.

Can you change the bot speed? How?

If we change Kp and Kd, what will happen?

Here is another version, line tracking with sensor calibration.

```
'Line tracking with again adjust
'computes error, adjust steering

SetPin gp0, PWM0A
SetPin gp1, PWM0B

SetPin gp10,DOUT  'right dir
SetPin gp11,DOUT  '*right dir

SetPin gp12,DOUT  'left dir
SetPin gp13,DOUT  '*left dir


Rem ch,frq,A,B

Sub forward dutyA, dutyB

 Pin(gp10)=1 : Pin(gp11) = 0
 Pin(gp12)=1 : Pin(gp13) = 0

PWM 0,500,dutyA, dutyB

  End Sub

Sub reverse dutyA, dutyB

 Pin(gp10)=0 : Pin(gp11) = 1
 Pin(gp12)=0 : Pin(gp13) = 1

PWM 0,500,dutyA, dutyB

  End Sub

Sub stop

 Pin(gp10)=0 : Pin(gp11) = 0
 Pin(gp12)=0 : Pin(gp13) = 0
```

```
 End Sub


Sub turnright dutyA, dutyB

 Pin(gp10)=0 : Pin(gp11) = 1
 Pin(gp12)=1 : Pin(gp13) = 0

PWM 0,500,dutyA, dutyB

 End Sub


Sub turnleft dutyA, dutyB

 Pin(gp10)=1 : Pin(gp11) = 0
 Pin(gp12)=0 : Pin(gp13) = 1

PWM 0,500,dutyA, dutyB

 End Sub

Sub line_status

 If A>1.2 Then Pin(gp3)=0
 If A<0.8 Then Pin(gp3)=1

 If B>1.2 Then Pin(gp2)=0
 If B<0.8 Then Pin(gp2)=1

 End Sub


Rem line tracking test

SetPin gp26,ain
SetPin gp27,ain

SetPin gp2,dout
SetPin gp3,dout
```

```
SetPin gp8,din,pullup
SetPin gp7,din,pullup

Kp = 1.5
Kd = 1.2

currentSpeed=20
gain = 0.62

start:

Do While Pin(gp8)=1

 line_status

 A = Pin(gp26)*gain
 B = Pin(gp27)

 err = (A-B)*5

 P = err
 D = err - last_err

 PD = P*Kp + D*Kd

 Pout = PD

 last_err = err

 Pleft = currentSpeed + Pout
 Pright = currentSpeed - Pout

  Print err,Pleft,Pright

  forward Int(Pleft),Int(Pright)

 Loop

 Stop
```

```
Do While Pin(gp7)=1
 Loop
  Pause (2000)
   GoTo start
```

This program provides gain adjustment for sensor A, gp26. The adjustment coefficient is found on the LAB 7 Line sensor. For this example, the error computed from the difference between sensor A and B is multiplied with 5. This adjusts the response of the bot for a deep curve.

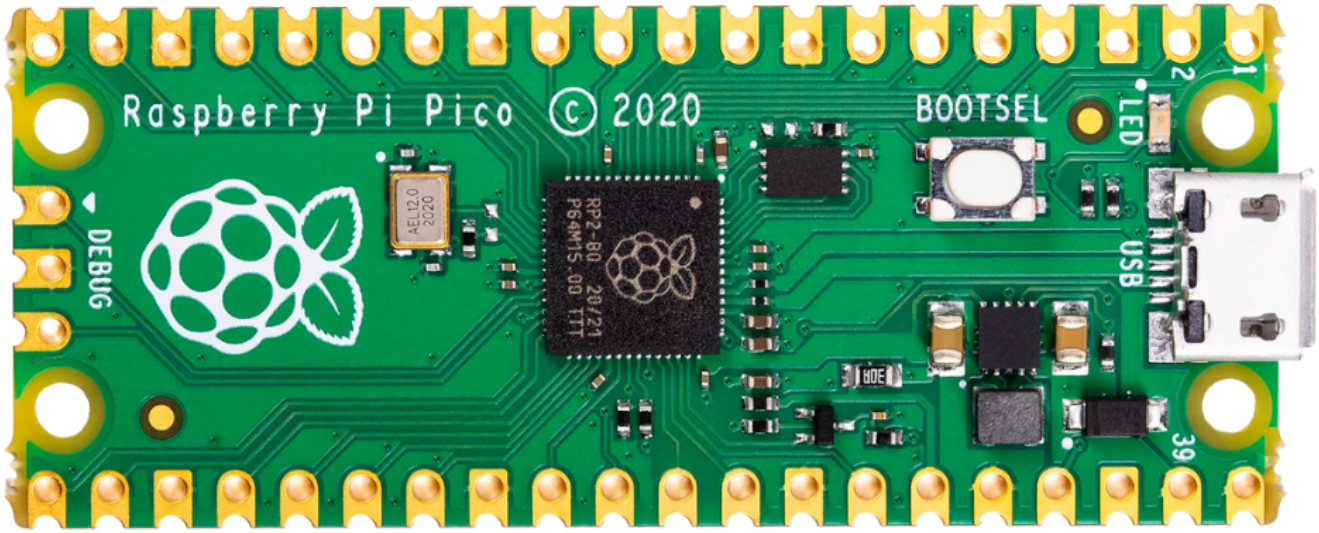
This limitation is from the use of only two sensors, left and right. In order to give the bot a higher response, the black line sensor needs to be of a higher resolution. Some higher-speed line tracking uses an array of line sensors.

Here is the video sample of the bot running.

https://youtu.be/TQvV-E3dKBo?si=e-au7MlEQoC_gJog

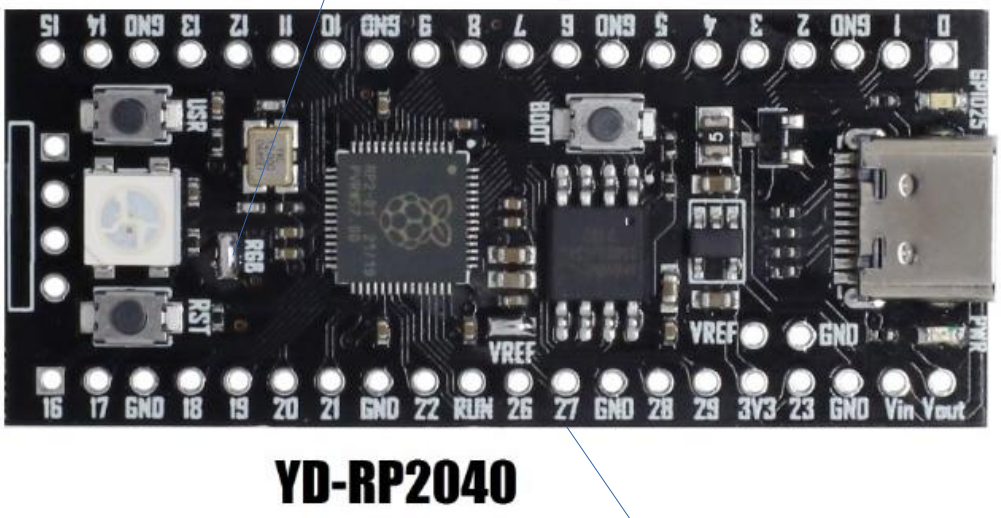
A. PI Pico controller board

The controller board for Picobot is the Raspberry Pi Pico, RP2040 MCU.



Another compatible Pi Pico is YD-RP2040 board. This board has RGB LED, at GP 23.

To use RGB LED, the solder pad RGB close to the LED must be soldered.



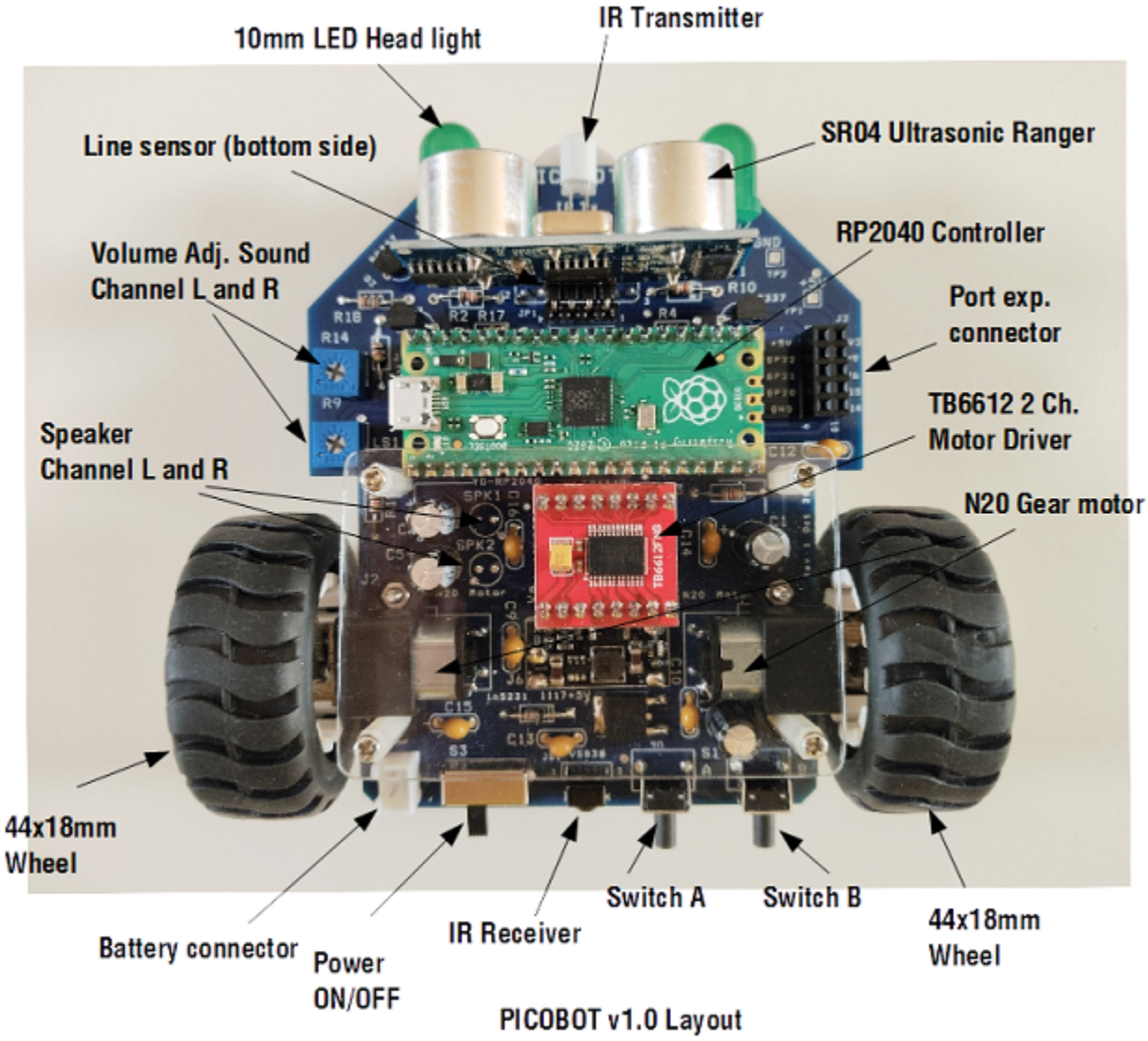
YD-RP2040

To use +3.3V Vref for the analog to digital converter, solder pad, VREF must be soldered.

To test RGB LED, MMBasic has function WS2812 for setting the color. Color format is 24-bit, Red, Green, Blue. Try change the value in hex number for each color.

```
>SetPin gp23, dout
>WS2812 o, gp23, 1, &H101000
```

B. PICOBOT v1.0 Layout



PICOBOT v1.0 Layout

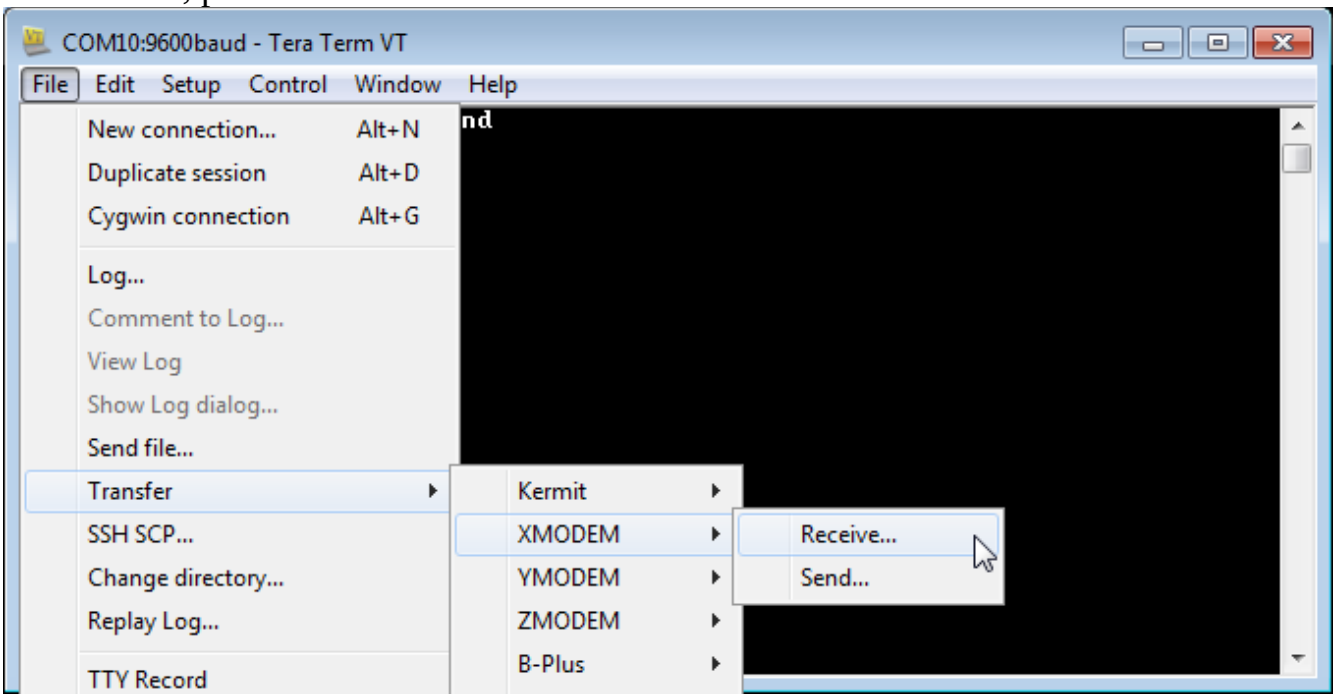
C. Save and Load BASIC program to/from HOST PC

BASIC program on the bot can be saved to the host PC with Binary file transfer protocol, XMODEM.

To save the program, press F12 to send the program to PC.

```
>
> XMODEM SEND
>
```

On terminal, press File>Transfer>XMODEM>Receive



Enter the file name be saved. The BASIC program on the bot will be saved on PC.

To Load BASIC program from PC to the bot, press F11.

```
>
> XMODEM RECEIVE
>
```

On PC, press File>Transfer>XMODEM>Send

Then select the file to be sent. The BASIC program from the host PC will write to the bot.