The Maze Runner - Tutorial

Introduction:

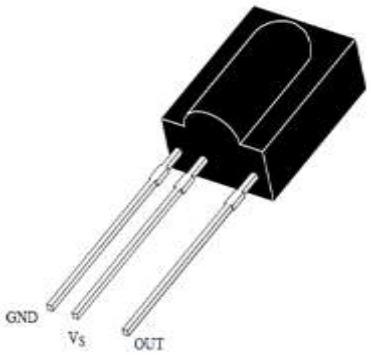
We know you can solve the problem statement by yourself but in case you are stuck about something we are introducing this tutorial for our event The Maze Runner. As per as our problem statement, this doc will cover different parts. First up is the section we think you are most likely to be stuck with.

IR Communication (What and how-to):

You have used remote controlled appliances, right? Well, most of them communicate using IR (infrared). An IR led transmits the light that the IR sensor receives and some sort of controller decodes it. However, IR communication is coded by means of duration of the on and off pulse. Decoding the pulse is manufacturer specific though. Here we are using Sony's SIRC protocol. We will talk about the protocol later. Let us jump right onto the implementation part.

Hardware:

To receive the signal, you need a modulated IR sensor, TSOP1738 or anything that works with 38 kHz modulated signal. If it's a little bit daunting then go with what we recommend. An example is given below…



GND: ground, V_s: +5v and OUT: digital out

Power it up correctly and connect the OUT pin directly to your microcontroller digital pin. One thing to note is that the TSOP1738 output pin is active low (low in presence of modulated IR, high in every other case).

To send an IR signal (if you want to test your receiver and we think you will) you need an IR LED. They look like normal clear-glass LED but emit light in the IR spectrum. Identify the pins correctly as you would with any other LEDs. Do not give more than 3V continuously otherwise you risk burning it. Remember you cannot see it with naked eye but your camera CAN.

Code:

Here we are assuming you are using Arduino and will give code examples specific to Arduino but if you are using another microcontroller, see note 1 below. Arduino has a very good library for handling IR communication, the IRRemote library (https://github.com/shirriff/Arduino-IRremote). Install it as per as the link. To receive and see IR codes, run the example "IRrecvDemo" or copy-paste the code from here…

```
#include <IRremote.h>

int RECV_PIN = 11;

IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{
	Serial.begin(9600);
	irrecv.enableIRIn(); // Start the receiver
}

void loop()
{
	if (irrecv.decode(&results))
	{
		Serial.println(results.value, HEX);
		irrecv.resume(); // Receive the next value
	}
}
```

Connect the TSOP OUT pin to Arduino DIGITAL_PIN 11 and upload this code to the board, open serial monitor and point any remote to the sensor and press a key, it should show a HEX code, for example 0x662A. If you've arrived at this point, the receiving and decoding part of IR communication is done. Pretty easy, wasn't it?

To send IR codes in Sony's protocol, connect the IR LED +ve pin to Arduino DIGTAL_PIN 9 and LED –ve pin to Arduino ground. After installing the same library, copy paste the code…

```
#include <IRremote.h>

IRsend irsend;

void setup()
{
}

void loop()
{

	for (int i = 0; i < 3; i++)
	{
		irsend.sendSony(0xa90, 12); // Sony TV power code
		delay(40);
	}
}
```

And press upload. After it is uploaded, point your phone's camera (any digital camera) to the IR LED and if you see the LED blinking it is working alright.

NOTE: You have to use two different Arduino to use the send and receive function simultaneously. One Arduino will not be able to run the codes even if you merge those two.

**NOTE 1: If you are using

1)an AVR, the IRRemote library uses avr timer interrupts. It uses one of the timers of AVRs, sets the timer counter overflow at 50ms and the interrupt service routine (ISR) checks the status of the IR sensor input pin. When it detects a start bit, it starts saving the durations when the pin change occurs and then decodes the duration to binary according to the protocol. Convert it to integer and you've got the IR code.

Wall Detection:

So, to follow a maze, you have to detect the walls successfully. There are basically two ways to do this, an IR proximity sensor and an ultrasound sensor.

Getting distance with IR sensor requires a pricey sensor like the Sharp branded sensors whereas ultrasound sensors (HC-SR04 used here) are pretty cheap and gives the distance with pretty good accuracy.

Hardware:

The ultrasound sensor HC-SR04 is pretty cheap (<200Rs.) and requires little configuration. It has 4 pins, the ever common VCC (+5V) and GND (ground) and the extra two pins, the TRIG and ECHO pins.

The TRIG and ECHO pins both go to your Arduino digital pins, let us assume you connected the TRIG pin to 11 and ECHO to 12.

Code:

Again we are assuming you are using Arduino but if you are using something else, this part is relatively easy.

YOU CAN NOT USE THE LIBRARY 'NEWPING' WITH IRREMOTE WITHOUT MODIFYING ONE OR BOTH LIBRARIES AS BOTH USE INTERRUPT SERVICE ROUTINE (ISR).

To send an Ultrasound, henceforth mentioned as US, you need to pull the TRIG pin high for 10microseconds, we will do it for 1ms. It then sends the signal in pulses and waits for it to reflect back. After it has received the signal, it pulls the ECHO pin high for the duration it took the signal to come back. So, if we get the time and multiply with speed of sound, we get the roundtrip distance. Hook up your Arduino, the ultrasound sensor, upload the code given below and open your serial monitor to view the distance of the object from the US sensor.

```
int ECHO_PIN = 12;
int TRIG_PIN = 11;
const int  cmsPerMs = 33.205; //may not be correct
void setup()
{
    Serial.begin(9600);  //start serial comm to  monitor the distance
    pinMode(ECHO_PIN, INPUT);
    pinMode(TRIG_PIN, OUTPUT);
}

void loop()
{
    digitalWrite(TRIG_PIN, HIGH);
    delay(1);
    digitalWrite(TRIG_PIN, LOW);
    double leftDistance = (pulseIn(ECHO_PIN, HIGH,
40000))*cmsPerMs/2000;    // gives distance in cms,, divided by 2 to get one trip
distance,
    Serial.println(leftDistance);
}
```