OpenCV - Maze detection

Main idea here is to determine which vertexes are connected to each other, get the lenght of the connecting lines and put those informations in a human readable file. First step can be for example find and enumarate those vertexes. Second step will be to examine them and determine whether they are connected. If this is is the case and they are connected, do not forget to remember the lenght of the line connecting them.

NOTE Opencv's function goodFeaturesToTrack(...) is a good place to start

OpenCV - Sign Detection

Think about the placement of the shields to minimize environmental interference. We will provide you with a few example pictures, so you can get to work right away.

You should try to make use of any features for a given sign - color, size, shape...

3.4. Sample maze generator

In order to be able to test your maze recognition and solving algorithms, we provided a maze generator. You can find it in **/home/pi/maze_generator**: running "./generate-mazes.sh n" generates n mazes in the "mazes" subdirectory. You can change the properties of the generated mazes in generate-maze.c using the defines at the top.

The demo maze has a size of 17x29 while the final maze has a size of 19x36.

3.5. Transmission Control Protocol/Internet Protocol (TCP/IP)

You may find it useful to divide the challenge into several programs which communicate via TCP/IP, or you may want to use one of the Off-the-Shelf Modules that offer a TCP/IP based protocol. Below, you can find code snippets illustrating how a TCP/IP connection can be established.

client.py

PYTHON

```
#!/usr/bin/env python

import socket

#connection to SERVER
TCP_IP = 'localhost'
TCP_PORT = 5003
BUFFER_SIZE = 1024
socket_client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

socket_client.connect((TCP_IP_CAM, TCP_PORT_CAM))

msg = "Hello!"
socket_client.send(msg.encode()) # messages are typically encoded into a bytearray to comply with python 3
server_answer = socket_client.recv(BUFFER_SIZE_CAM) #blocking
server_answer = cam_answer.decode()
print(server_answer) #prints "Hello back!"
```

server.py

PYTHON

If you have tested the data transmission between the 3pi Robot and the Kubuntu VM successfully and want to incorporate your data exchange protocol on the Raspberry Pi using python, you can use code like this (taken from here (https://pythonhosted.org/pyserial/)):

PYTHON

```
import serial

ser = serial.Serial('/dev/ttyS0', 115200)
ser.open()
ser.write(b'Hello World')
character = ser.read()
print(character)
ser.close()
```

Please note that while the Kubuntu VM uses /dev/ttyUSB0, the Raspberry Pi uses /dev/ttyS0!

4. Off-the-Shelf Modules

A lot of work will be necessary to solve the challenge, so just like in the real world, you can decide to outsource the development of some software modules for a cost. However, the modules that have been created use fixed protocols and APIs, so you don't have a say in which protocol or API you'd like. That's the tradeoff for being able to use them immediately :)

Regardless, you may consider to use some of these to avoid the risk of not having a working system when the race starts. For this reason, it might make sense to use similar protocols and/or APIs so that you can easily incorporate an off-the-shelf module if you find that you can't complete the module in time yourself. Modules can be bought up until 1 hour before the final challenge.

When you decide to purchase one of the modules, you will receive the source code of the module. For this reason you're not able to return the module afterwards, the purchase will be final.

To understand how each of the modules that you can buy fits into the system, you can use our system architecture as a reference:

```
count - 94
start - 91
end - 92
0-7,22,2
0-11,21,-2
0-39,92,1
0-64,22,-1
1-44,25,2
1-48,22,-2
1-81,47,1
1-82,24,-1
2-3,24,-2
2-71,116,-1
2-92,43,2
3-35,66,1
4-35,19,2
4-59,20,-1
5-62,19,1
6-7,19,-1
6-36,19,2
8-38,43,1
8-59,20,2
8-60,23,-2
9-10,19,-1
9-62,43,2
10-39,19,2
...etc...
```

4.2. Robot Control Module

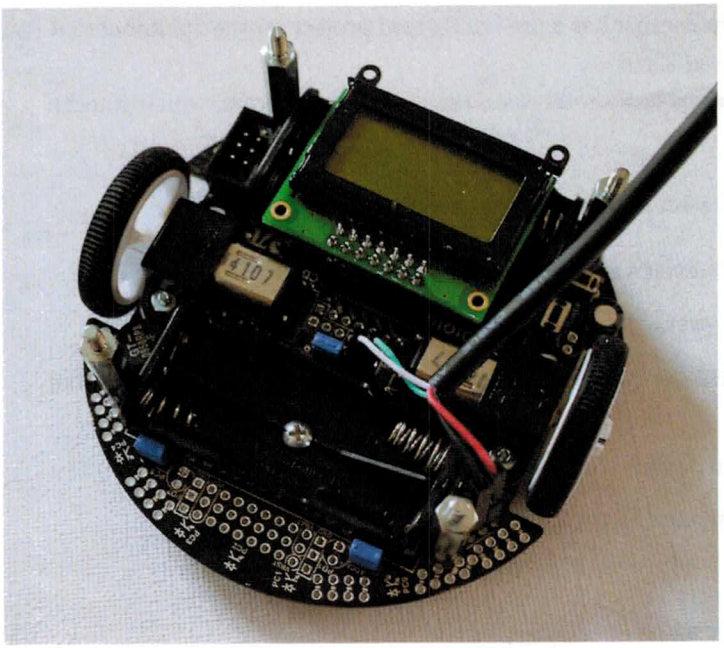
If you decide to buy the robot control module, you will receive an Eclipse project containing the source and build configuration for the 3pi Robot firmware that we use. The robot then behaves in the following fashion:

- After a reset, it will wait for the user to press the B button on the robot.
- After the B button has been pressed, it will calibrate its brightness sensors and then listen for commands sent over the UART interface.
- When the ASCII character '**L**' was received, the robot will rotate 90 degrees to the **left**, then respond with a '**D**' and wait for the next command.
- When the ASCII character '**R**' was received, the robot will rotate 90 degrees to the **right**, then respond with a '**D**' and wait for the next command.
- When the ASCII character '**U**' was received, the robot will rotate 180 degrees (**turn around**), then respond with a '**D**' and wait for the next command.
- When the ASCII character '**F**' was receievd, the robot will drive **forward** until it encounters an intersection, then respond with a '**D**' **followed by a combination of** '**F**', '**L**' **and/or** '**R**' (depending on which roads exist at this intersection) and wait for the next command.
- When the ASCII character '**B**' was received, the robot will **beep**, then respond with a '**D**' and wait for the next command.
- While driving forward, the robot stops when the line is not recognized any longer.

4.3. Road Sign Detection Module

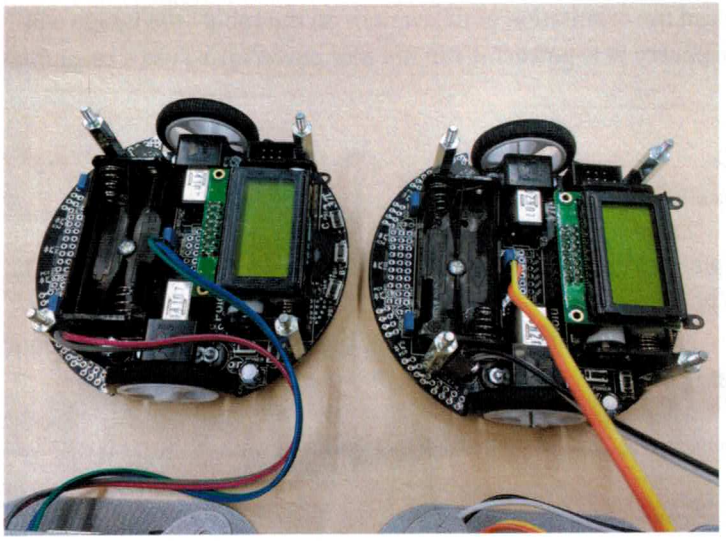
If you decide to buy the road-block detection module, you will receive a compressed archive containing the module's python source ready to use on the Raspberry Pi. The module acts as a service and behaves in the following fashion:

- When started, the module initializes the camera using OpenCV and begins listening on TCP port **5003** for incoming connections.
- When a connection has been established, it waits for incoming requests.
- When the character string "**check**" is received, it will capture a frame from the camera, perform object detection and write



To reconnect the 3pi Robot and the Raspberry Pi, perform the following steps:

1. Ask all your team members if they are working on the Raspberry Pi right now. If anyone is, you cannot proceed!
2. Shut down the Raspberry Pi from the desktop menu
3. Disconnect the battery pack from the Raspberry Pi
4. Disconnect the USB/UART interface from the 3pi Robot
5. Attach the Raspberry Pi assembly to the robot base
6. Reconnect the 4 cables that connect the 3pi Robot with the Raspberry Pi (see picture below)
7. Reconnect the battery pack to the Raspberry Pi or use a USB cable to supply power



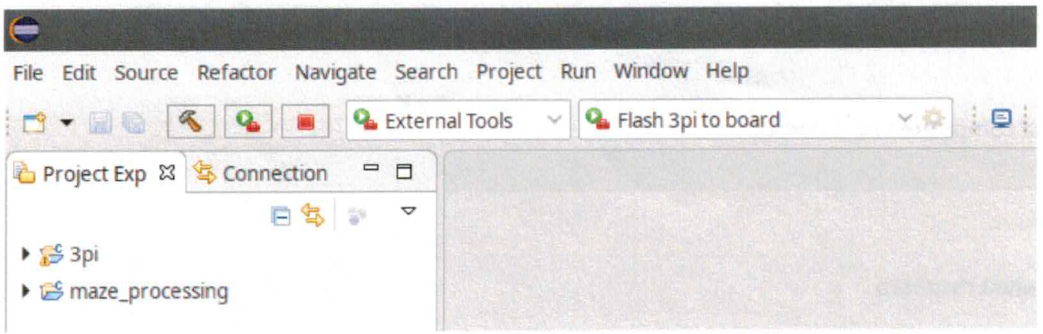
3.3.2. Firmware Setup

There are three ways to create firmware for the 3pi Robot:

a) Use Eclipse (reference project in **/home/pi/3pi/eclipse-workspace**)

b) Use the Arduino IDE (run "arduino"; 3pi Robot library installed in /home/pi/sketchbook/libraries; see https://www.pololu.com/docs/0J17; no reference sketch available)

c) Use a makefile-based project (in **/home/pi/3pi/makefile_project**; same content as the Eclipse project)



You are now ready to work on either project.

3.3. Working with the 3pi Robot

The 3pi Robot is a robot platform with integrated sensors, motors and an Atmel ATmega328P microcontroller. The manufacturer provides an Arduino-compatible library to simplify access to the peripherals. For this challenge, we only need to use the 5 line sensors, the 2 motors and the buzzer. You can find the original documentation on our storage medium or at the following URLs:

- 3pi users guide: https://www.pololu.com/docs/pdf/0J21/3pi.pdf
- 3pi quick start: https://www.pololu.com/file/0J122/3pi-quick-start.pdf
- 3pi simplified schematic: https://www.pololu.com/file/0J119/3pi-schematic.pdf
- 3pi AVR library command reference: https://www.pololu.com/docs/pdf/0J18/avr_library_commands.pdf

3.3.1. Hardware Setup

As the 3pi Robot is a base platform for the Raspberry Pi, both can be separated. Doing so allows you to work on both parts individually.

To separate both, perform the following steps:

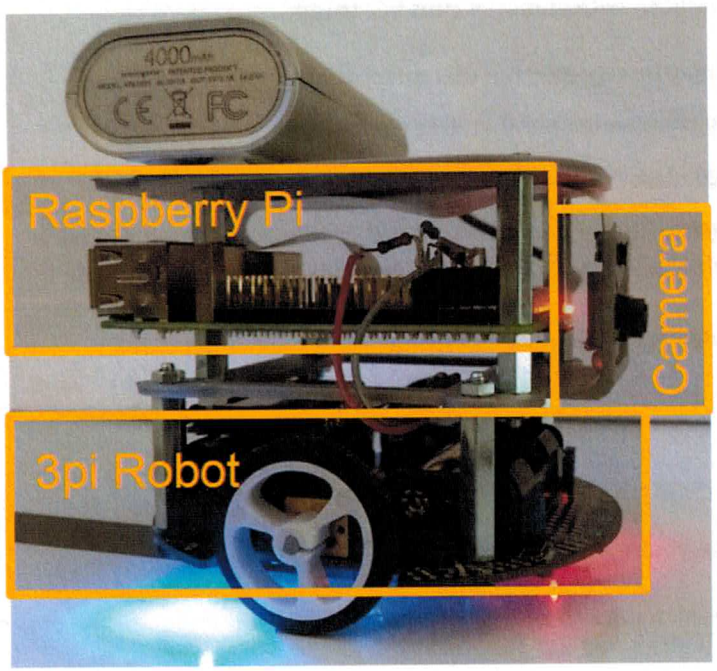
1. Ask all your team members if they are working on the Raspberry Pi right now. If anyone is, you cannot proceed!
2. Shut down the Raspberry Pi from the desktop menu
3. Disconnect the battery pack from the Raspberry Pi
4. Disconnect the 4 cables that connect the 3pi Robot with the Raspberry Pi at the robot base
5. Remove the Raspberry Pi assembly from the robot base
6. Reconnect the battery pack to the Raspberry Pi or use a USB cable to supply power
7. Connect the USB/UART interface to the 3pi Robot:
   - Red = +5V (charge port on the 3pi Robot)
   - Black = GND (charge port on the 3pi Robot)
   - Green = TXD (RXD on the 3pi Robot, which is PD0)
   - White = RXD (TXD on the 3pi Robot, which is PD1)

The connection then looks like this:

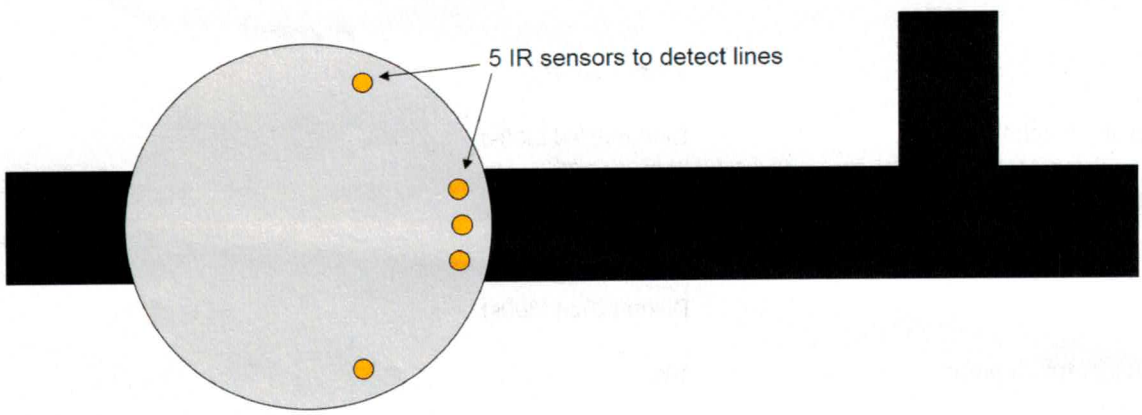
3. Quickstart Guidelines

We do want you to focus on solving the challenge and not waste too much time setting up tools and digging into manuals to learn how to program a robot, how to develop for the 3pi Robot, Raspberry Pi, et cetera. Therefore, this chapter provides you with some quickstart hints and a virtual machine image containing a toolchain setup that you can use to get started quickly. Please also don't hesitate to ask your team supervisor for assistance when you need it.

Each team gets their own RoboCar for testing and preparing for the final challenge. The RoboCar consists of a 3pi Robot that has a Raspberry Pi mounted on its top. At the front of the RoboCar a camera is mounted that is directly attached to the Raspberry Pi. The 3pi Robot and the Raspberry Pi communicate via a UART interface.



The 3pi Robot has two wheels driven independently by two motors allowing it to rotate around its current position. A small ball mounted on the backside makes sure that the 3pi Robot is not tipping (at least not to its back). The 3pi Robot is equipped with 5 infrared brightness sensors on its bottom. These can be used to detect and follow a black line drawn on the ground. It's a good idea to stop when the 3pi Robot lost track of a line to avoid risking any damage due to uncontrolled movements.



3.1. Working with the Raspberry Pi

On top of the 3pi Robot sits a Raspberry Pi of the third generation. It has the following features:

- 1.2GHz 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN, Bluetooth 4.1 and Bluetooth Low Energy (BLE)
- 4 USB ports
- 40 GPIO pins