Cruise control Project

User Manual

Crtl Alt Elite :

Eunhye YOON - Piyuan SONG - Abishek Anil MUDALE - Alexandre LIN - Adrien SALLÉ

Teachers :

Ms LETOURNEUR – Mr PERRIER

# Introduction

This project uses modular design and manufacture cruise control system. The modular system is provided by Lego. The project is divided into three parts: bronze, silver and gold. The bronze goal will enable the car to find its own way and monitor obstacles, while the silver goal will enable the car to navigate more complex routes, with the ultimate goal being that the car can detect roadside panels. This article will provide you with information about Lego, sensors and how to use the software.

Table of content

[Introduction 2](#_Toc124941429)

[1. Requirements 4](#_Toc124941430)

[1.1. Cruise control 4](#_Toc124941431)

[1.2. Line Following and Map 4](#_Toc124941432)

[1.3. Assembly of robots 5](#_Toc124941433)

[2. Hardware 6](#_Toc124941434)

[2.1. Frame of vehicle 6](#_Toc124941435)

[2.2. Selection and installation of sensors 6](#_Toc124941436)

[2.3. Connection between ev3 and sensor 6](#_Toc124941437)

[3. Software 8](#_Toc124941438)

[3.1. Integrated development environment 8](#_Toc124941439)

[3.2. Ev3dev library 8](#_Toc124941440)

[4.Operation procedure 10](#_Toc124941441)

[4.1. Setting up the Linux environment on the ev3dev 10](#_Toc124941442)

[4.2. Write and upload code 10](#_Toc124941443)

[4.3. Track making and road signs 10](#_Toc124941444)

[4.4. Launching code 11](#_Toc124941445)

[4.5. Code structure 12](#_Toc124941446)

[Tables of figures 15](#_Toc124941447)

# Requirements

## Cruise control

桌子上放着照相机

中度可信度描述已自动生成With the use of electronic technology, the driver need not control the accelerator pedal, and can ensure that the car runs at a pre-set speed. A car equipped with this device on the highway can save the driver from stepping on the accelerator frequently, hence reducing the driver fatigue level. Moreover, it improves the stability, safety and fuel utilization.

Figure : Cruise control vehicle

The project is based on the LEGO EV3, designed and produced by CtrlAltElite team.

## Line Following and Map

Different objectives require different test routes, so we need to use the drawing tool to create the required map. Use a simple drawing tool and split the map into different parts so that you can use it multiple times and splice it into different maps.

图示

描述已自动生成

Figure 2 : Map sample

Use tape to paste the map on the back, and put cardboard on the bottom to ensure that the map is smooth, so that the test section is more accurate.

Test only need to put the sensor and the route to it, start the car on the computer to complete the route test.

## Assembly of robots

The Lego EV3 box contains a simple assembly process book, EV3 series sensors, a controller and numerous parts. Select some of the parts to complete the project. It is divided into hardware part and software part, which will be explained in the following sections.

图形用户界面, 网站

描述已自动生成

Figure : Lego ev3 parts

# Hardware

## Frame of vehicle

Lego EV3 comes with a handy build-in manual. Following the manual, we can build a complete and simple vehicle. We placed two motors on each side of the vehicle that acts as driving motors and power the Lego car. The EV3 bricks are connected at the top to communicate with the vehicle, and are kept intact with the help of different Lego parts and connectors.

Diagram, engineering drawing

Description automatically generated

Figure : EV3 Robot

## Selection and installation of sensors

In total, 5 sensors are used. The selection of these sensors is done by the means of their properties. 3 colour sensors are used to detect the black line and follow the path upfront. At a slight elevation above the colour sensor, an ultrasonic sensor is connected to detect the obstacle coming in front of the vehicle. A push sensor is used to guide the vehicle to start and stop which is placed conveniently next to the Lego EV3 brick.

## Connection between ev3 and sensor

The connection is easy to implement with the help of various Lego pieces which have unique shape and dimensions and small connector joints. This makes a Lego vehicle not only simple to construct but also gives freedom to construct a complex model. In this vehicle the car is constructed in a way that the EV3 brick is faced up which makes it easy to handle and operate alongside a push sensor and three color sensors placed at the front at equal distance from each other and an ultrasonic sensor placed just above the color sensors to detect the obstacles. These sensors and EV3 brick are mutually connected by means of a multiplexer placed behind the Lego EV3 brick.

图示

描述已自动生成

Figure 5 : EV3 sensors and actuators connections

In most cases, the sensor interface provided by the EV3 controller is adequate. If the number of sensors required exceeds the number of interfaces, a multiplexer is required. The multiplexer can provide three additional interfaces, allowing access to more sensors. Just connect the main head to the EV3 controller and connect the additional sensors to C1 C2.

桌子上摆放着黑色的机器

低可信度描述已自动生成

Figure 7 : Lego multiplexer

# Software

## Integrated development environment

ev3dev is the operating system of the EV3 brick. This is a Linux environment. To create our code, we used Visual Studio code, which allows us to do autocompletion and basic spell checking as we code.

Une image contenant texte, capture d’écran, moniteur, noir

Description générée automatiquement

Figure 6 : Screenshot of Visual Studio Code

Visual Studio code can be searched and downloaded using google, and the tutorial can be viewed on the official website. Create a Linux code environment in the software and edit it.

Along with the extension ev3dev-browser for Visual Studio Code, we are able to send our program to the board through ssh.

## Ev3dev library

To ensure the stability of our robot, we decided to use the PID control technique. This method is widely used in the robotic domain and has the advantage of being efficient and quick to set up, although it might be difficult to understand for beginners.

The following example is the PID block diagram required for the silver target. You can adjust values of PID controller (kp, ki, kd) depending on your environment.

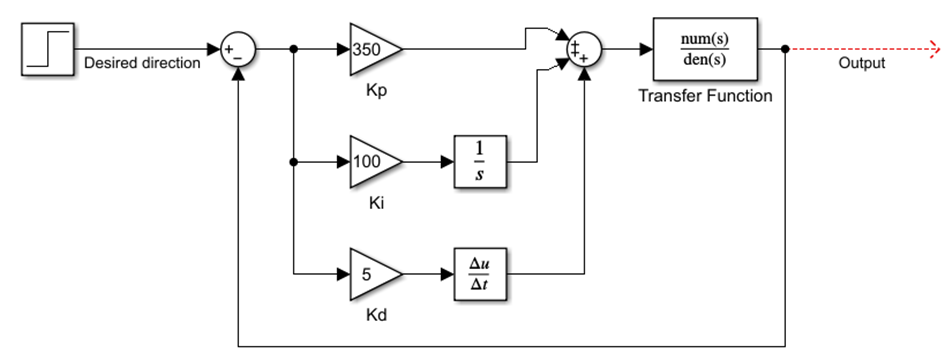


Figure 7 : PID control schematic on SIMULINK

# 4.Operation procedure

## 4.1. Setting up the Linux environment on the ev3dev

Before this step make sure to follow the tutorial to setup the Linux environment and, the network and SSH connection. (Link here: <https://www.ev3dev.org/docs/getting-started/>)

For simpler processes it is recommended to use Visual Studio code and the ev3dev-Browser add on. (Documentation here: <https://marketplace.visualstudio.com/items?itemName=ev3dev.ev3dev-browser>)

## 4.2. Write and upload code

After setting up your ev3dev block, you can now start writing code for it. An example of code can be found on this repository: [*https://github.com/AlexandreLin888/Isep\_cruise\_control*](https://github.com/AlexandreLin888/Isep_cruise_control)

Once the code is done, send it to the ev3dev by clicking on “Send project to device”

Your project should then be visible on the ev3dev Linux interface.

## 4.3. Track making and road signs

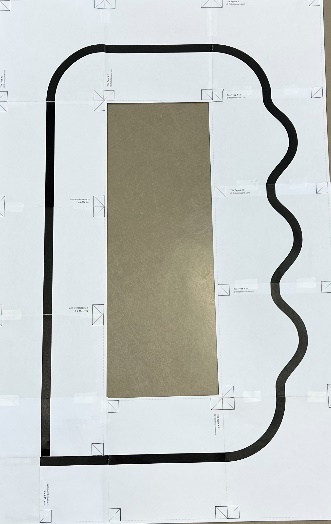
Since the robot is a line following robot, it needs a track to run on. The track can be built with track part that can be found on this website: <https://robotsquare.com/2012/11/28/line-following/>

Figure : Track example

Once the track is printed, it is advised to glue the track on a cardboard or something similar to have as smooth path and increase sensor precision

If a pixy2 camera is used, it is possible to create some road sign that can be recognized by the camera with barcodes. The barcodes can be found here:

<https://raw.githubusercontent.com/charmedlabs/pixy2/master/documents/other/all_codes.pdf>



Figure : Panel examples

## 4.4. Launching code

We launch the code through an ssh terminal. It allows us to manage the files on the robot and launch every program separately.

Une image contenant texte

Description générée automatiquement

Figure 10 : SSH console (Powershell)

To navigate through the linux console, we use the command “ls” to display the folders content and the command “cd [PATH]” to go from one folder to another.

Launching a code will be done by using python with the following command :

sudo python3 [programFile]

Here, “sudo” isn’t mandatory, however we recommend using it as some system resources used by the programs may be protected.

## 4.5. Code structure

The code works with three files :

* *main.py* : The file to be executed, with the main state machine and the threads initialization.
* *robot.*py : The file were all the functions and the threads are defined, it is also were the robot settings are defined.
* *pixy2.py* : The file with the pixy2 class, allowing us to use the camera

Here is a summary of the code classes :

Une image contenant texte

Description générée automatiquement

Figure  : Class summary

The program run as follow :

* At launch, the main program will reference the robot.py and the pixy2.py files. It will then initiate two threads, a camera thread, and a ultrasonic sensor thread. After that the program will enter a state machine in an infinite loop in which it will run a PID control algorithm to follow the line.
* While the program is running, the camera thread will send request using I2C bus to enquire if a panel (in the form of a barcode) has been seen. If the camera detect a panel, it will return an array filled with the number associated with the panels on the image. Depending on the panel seen, the thread will change the value of the ‘stateIndex’ variable which will be read in the main thread to change state in the state machine.
* The sensor thread purpose is to read the ultrasonic sensor value and compute a correction depending on the measured value. This correction is read at each loop occurrences by the main thread to adapt the motor power and stop if needed the robot.

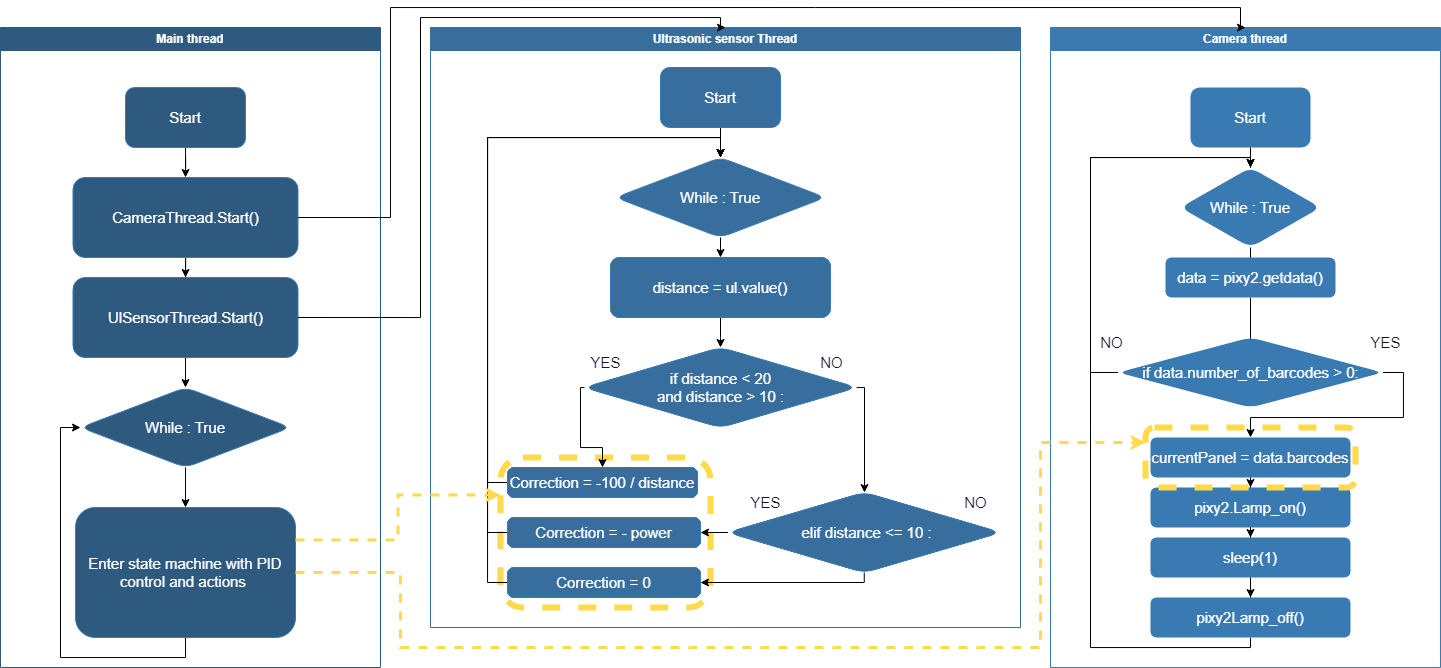


Figure : Schematic of the program threads

The state machine of the main program works depending on the panel seen by the camera. When a panel is seen, the state will change and the robot will execute the associated action before going back to the PID control state. The states are defined as follow :

Une image contenant texte, ipod

Description générée automatiquement

Figure : State machine of the program

# Tables of figures

[Figure 1 : Cruise control vehicle 4](https://iseparis-my.sharepoint.com/personal/euyo62166_eleve_isep_fr/Documents/Microsoft%20Teams%20Chat%20Files/ctrlaltdel%201.docx#_Toc124941448)

[Figure 2 : Map sample 4](#_Toc124941449)

[Figure 3 : Lego ev3 parts 5](#_Toc124941450)

[Figure 4 : EV3 Robot 6](#_Toc124941451)

[Figure 5 : EV3 sensors and actuators connections 7](https://iseparis-my.sharepoint.com/personal/euyo62166_eleve_isep_fr/Documents/Microsoft%20Teams%20Chat%20Files/ctrlaltdel%201.docx#_Toc124941452)

[Figure 6 : Screenshot of Visual Studio Code 8](#_Toc124941453)

[Figure 7 : PID control schematic on SIMULINK 9](#_Toc124941454)

[Figure 8 : Track example 10](https://iseparis-my.sharepoint.com/personal/euyo62166_eleve_isep_fr/Documents/Microsoft%20Teams%20Chat%20Files/ctrlaltdel%201.docx#_Toc124941455)

[Figure 9 : Panel examples 11](https://iseparis-my.sharepoint.com/personal/euyo62166_eleve_isep_fr/Documents/Microsoft%20Teams%20Chat%20Files/ctrlaltdel%201.docx#_Toc124941456)

[Figure 10 : SSH console (Powershell) 11](#_Toc124941457)

[Figure 11 : Class summary 12](#_Toc124941458)

[Figure 12 : Schematic of the program threads 13](#_Toc124941459)

[Figure 13 : State machine of the program 14](#_Toc124941460)