AutoPylot

Maxime Ellerbach
Mickael Bobovitch
Alexandre Girold
Maxime Gay

Group: Autonomobile

Contents

1 Introduction

1.1 Project nature

Autonomous cars will likely dominate our roads in a relatively close future. with this project, we aim to have a first approach to such a complex problem by participating in the DiyRobocars competition. We will be using a 1:10 scale radio controlled car modified to integrate onboard computer such as a raspberry pi and some sensors.

1.2 State of the art

In this section, we will try to see what was previously made in this sector of industry. It would not be realistic to compare our 1:10 project to real sized cars such as Tesla's, simply because in a racing environnement, we don't need to deal with such an amount of safety: pedestrian detection, emergency braking, speed limit detection and other. So we will only see miniature autonomous racing framework that we would likely race against.

The most known is called "DonkeyCar", created by Will Roscoe and Adam Conway in early of 2017. Most of the models trained with DonkeyCar are behavior cloning models, meaning models that tries to replicate the behavior of a driver. This methods uses a big amount of images (input) associated to steering angles and throttle (output), it requires the user to drive the car (collect data) prior to training the model: no examples means no training. The lack of training data often leads to the car leaving the track.

One other framework worth looking at is one created by Nvidia called "JetRacer" released in 2019. It uses a different approach from DonkeyCar where the user annotates the images by hand by clicking on where the car should go. The model used is similar to what DonkeyCar uses meaning a Convolutional Neural Network with one input (the image) and two outputs, one for the steering angle and one for the throttle to apply.

Both those framework are written in python and use packages such as Tensorflow and OpenCV, will also use them in our project.

2 Objectives

2.1 Final objectives

Our main objective is to make our car race against other cars and win the race ! This will require multiple intermediate milestones:

- Being able to send scripted controls to the motor and servo.
- Being able to drive the car manually using a controller.
- Develop a way to gather images and annotations and store them in a stuctured way, for example sorted by date.
- Process those data before feeding them to the neural network.
- Being able to train a convolutional neural network using those data.

- Tweak the architecture and the parameters of the chosen model to acheive the best results.
- Test in real life the model.
- Race against other !

Once all of that is done, we will start our optional objectives that will enable better racing and better understanding of the car's environnement.

2.2 Optional objectives

To be able to go faster and increase the reliability of our car's driving, we will need to add some features to our project.

One good feature would be to have a model that takes into account the speed of the car. As we all know on a real car, we don't steer the same way when going at 10km/h and going at 90km/h. This input extension could bring more stability to our model. To go faster, it would also be great to be able to differenciate turns from straight lines and even braking zone before the turn.

One of the challenges racing implies is overtaking. Overtaking is a very complex manoeuver. First we obviously need to be faster than the car ahead, we could assure that by detecting the successive positions of the car ahead on the image. If the car is getting closer, it's Y coordinate will come closer to the bottom of our image. Also, we cannot overtake if we have no room for that, that means we have to detect a gap on the left or the right of the car before initiating the overtake. Once we decided on what part of track we will overtake, we still need to get ahead of the opponent's car. We could acheive that by forcing the model to drive on the left or right part of the track for a given amount of time.

2.3 Motivations

The concept of an autonomous car is nothing new. Indeed, many achievements have been made. Both real and miniature size cars exist but their implementation is different. In fact, there is a huge gap of technologies between. We want to reduce this gap. We want to succeed in implementing the safety and stability of cars in productions like Tesla. Our goal is not to make a miniature Tesla, but to take back the safety elements. It is common for miniaturized cars to slip off the tracks for lack of environmental detection. We want to use the latter method for entourage detection. So far, a large part of the implementations uses line detections. We want to go beyond the comfort zone and use a different and innovative method that is based on AI. We want to take the best of what is out there and explore possibilities to make self-driving miniature cars that are reliable and capable of racing other vehicles. Until now miniaturized cars did not take into account other vehicles. They are considered the only vehicle on the track and the overtaking principle was not implemented. We will try to beat this challenge. Indeed, this is something unusual. We want to make a real leap forward in this area and not just replicate what is out there. To simplify our task, we will be content to run races. The recognition of panels and pedestrians is too complex for us. Large-scale racing and artificial intelligence are a little exploited field. This is what push us to create something new. We will be happy to represent EPITA at the next cars races and win for our School. We would be very proud of it.

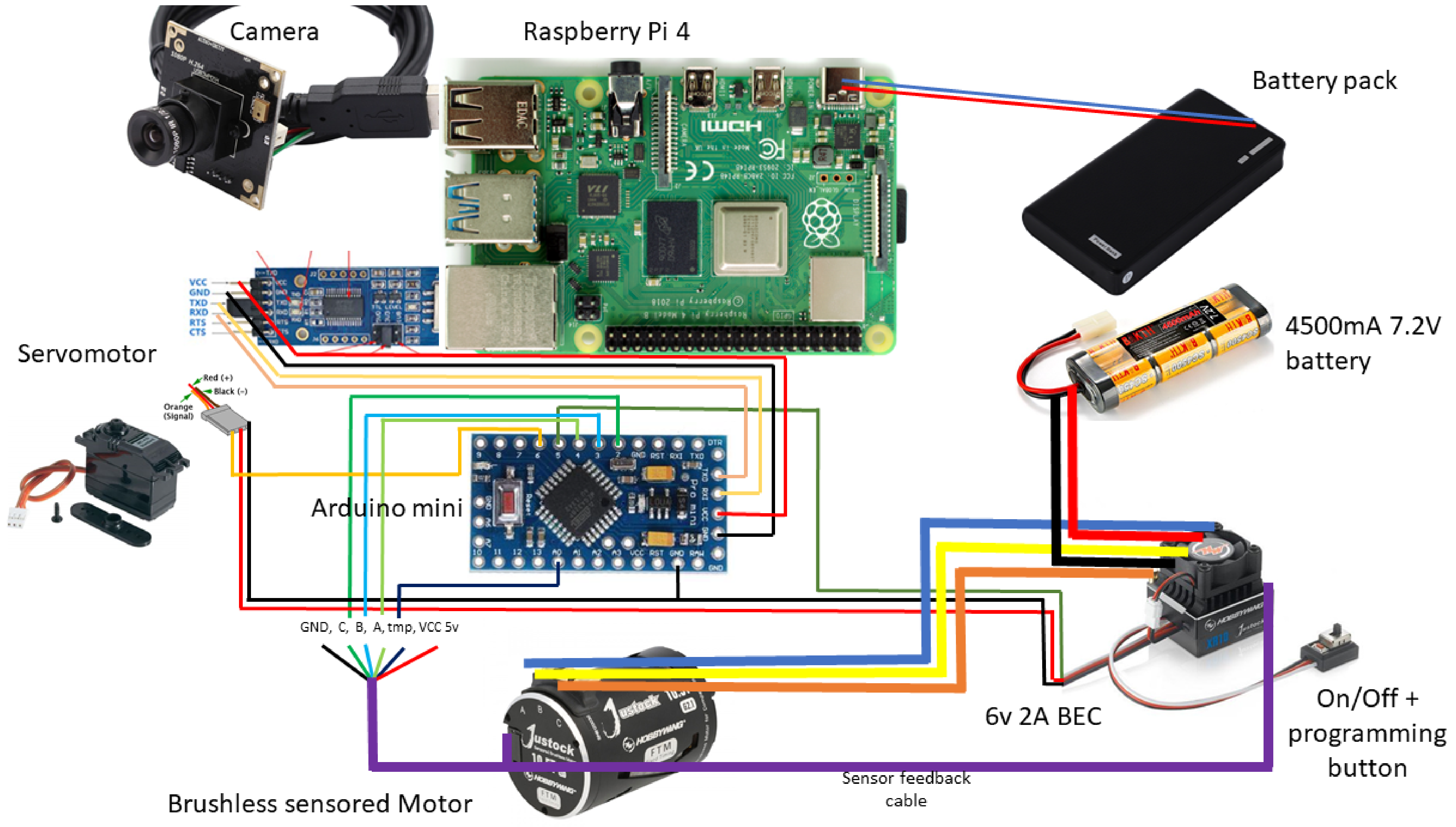
3 Technical specifications

3.1 Hardware

On the hardware side, we already have a working car containing:

- A RaspberryPi 4. It is used for heavy computations like image processing, model inference, and other. Most of our programs will run on this device.
- A USB camera. Connected to the RaspberryPi, this camera will be our main source of data.
- An Arduino Mini. It is used for the low level, it handles commands sent by the RaspberryPi on the serial port, processes them and send PWM signals to both the ESC (Electronic Speed Controller) and servo motor.
- An Electronic Speed Controller (or ESC). It is used to drive the motor, it receives from the arduino a PWM signal (Pulse width modulation).
- A Servo moto. Just like the ESC, it receives a PWM signal
- A Speed sensor. This sensor will be usefull for our optional objectives that will require a speed feedback. This sensor is read by the arduino, then the data is sent to the RaspberryPi over the serial port.

As the car is already working on the hardware side, it will save us some precious time. But still, it is really important to understand how the whole car is working and how the different components are interecting. Here is a schema of the current hardware setup we will be using:



Camera
Raspberry Pi 4
Battery pack
VCC
GND
TXD
RXD
RTS
CTS
Servomotor
Red (+)
Black (-)
Orange (Signal)
4500mA 7.2V battery
Arduino mini
GND, C, B, A, tmp, VCC 5v
6v 2A BEC
On/Off + programming button
Brushless sensored Motor
Sensor feedback cable

3.2 Software

As you understood, our work will be focused more on the software side. Our project will be written mostly in Python, with a bit of JS and arduino code. We will divide the project into 3 big parts:

- The backend. This is where most of our efforts will be focused, this part includes every key programs all the way from the data gathering to the model training and testing.
- The Telemetry website. This part will give us the ability to see in real time the last predictions of our model and plot some data found usefull like the speed of the car.
- The presentation website.

3.3 Constraints

- weight - language: python - autonomous navigation - data ? - framerate, efficiency of the algorithm - safety

4 Planning

Race 1 (29/01)	Race 2 (26/02)	Race 3 (19/03)	Race 4 (16/04)	Race 5 (14/05)	Race 6 (04/06)
0	1	2	3	4	5
6	7	8	9	10	11

5 Task allocation

Tasks	Mickael B.	Maxime G.	Alexandre G.	Maxime E.
Do Something	x			x
Do Something else		x	x	

6 Conclusion