

# AGV Car Project

---

This repo contains the resources for the AGV project. The goal of the project is to create a self-driving car, which uses a neural network for driving. The project has many approaches during the semesters, this document aims to collect, describe the project for future development.

Sections:

- Architecture
- Microcontrollers
  - ESP
  - Nucleo
- Python Server
- Neural Networks
- Simulation
- Issues
- Future Tasks

## Dictionary

---

Reward - Reward is calculated based on the network's decision State - State of the vehicle (position, speed, sensor data, image, decision of network, reward)

## Architecture

---

The system consists of three main components:

- Nucleo Controller
- ESP32-CAM Controller
- Python Server

The Nucleo board is responsible for handling sensor information, running the neural network and controlling the servos of the car. It works as a Slave to the ESP32-CAM Master. They are connected through SPI. The ESP32-CAM is responsible for communication between the car and the python server. It is also responsible for taking the image and sending the image to both sides of the communication. The Python Server is a socket based application, which waits for the connection of the ESP32-CAM client and sends commands and receives data gathered from the two controllers. After gathering data, the server trains a Neural Network, and sends the network to the Nucleo Controller through the ESP.

## Microcontrollers

---

### ESP

ESP is the communication master. All data flows through here between the server and Nucleo. There are currently two versions for the code:

- ESPRESS-IDF VS-CODE
- Arduino IDE

Both of them are built similarly, but each of them was incapable for SPI communication, more on this later. The structure of the code and workflow is that the ESP connects to the socket and sends out a request. These requests include:

- WAITING\_FOR\_COMMAND - This indicates to the server that the controller is waiting for action to perform.

Then based on the response of the server the ESP handles received actions.

Receives actions can be:

- STOP - Send stop message to Nucleo and stop all tasks
- LINE\_FOLLOW - Start a pre implemented line following algorithm on nucleo
- SEND\_IMAGE - Sends the image to the server
- RECEIVE\_WEIGHTS - Prepare for incoming weights from the server
- USE\_NETWORK - Run the network on the nucleo, and begin data transfer between server and nucleo(image, weights, states)
- NO\_NEW\_COMMAND - There has not been new input on server, continue as before

The controller uses the camera to create images for the neural network and send them to the server. Due to computational capacities the image is grayscale. The smallest available configuration is 96X96. Information about this we could not find on the github, but in the ESPRESS-IDF we get this size as a suggestion. More information about camera configuration can be found on the ESP32-CAM github. Definitions and values can be found in the header file.

We have set SPI communication between the two controllers. The table below will list the pin layout that is currently used.

Name	PIN
MOSI	12
MISO	13
SCLK	15
SS	14
HANDSHAKE	2

Handshake was used when the Nucleo was a master controller for signalling that the ESP is ready for communication.

## Development

Currently there are two alternatives to begin development with. First one is a C framework, ESPRESS-IDF ([link](#)). The second is the Arduino IDE ([link](#)). These projects both implement the same thing, configuring and fault detection is easier on the Arduino IDE. To be able to connect to a wifi network it is advised to use a mobile hotspot since the local wifi's setting doesn't allow hosting (internet is not needed, just the network). During development, the reset button has to be pressed on the controller to begin code transfer. This doesn't work all the time. For 100% success we have used putty to connect to the serial with rate 115200 and press reset.

## Arduino IDE

Code can be found [here](#).

The main difference is that this code uses a serial monitor, accessible through browser. This was helpful in configuration as when the ESP is connected to the Nucleo there is no access to the serial manually. Otherwise packages can be installed on the Arduino faster, and there are some convenient wrappers.

For socket communication the password and host IP can be set in the *wifi\_config.h* file.

Issues:

- The ESP SPI.h library is limited in data transfer. We were only able to send a 1 to 4 bytes of data. This is too slow to transfer the network weights
- Documentation and resources are hard to find for the ESP specific functions

## ESPRESS-IDF

Code can be found [here](#).

This framework provides robust development. It contains lots of examples and is well documented ([link](#)).

For socket communication the password and host IP can be set in the *sdkconfig* file.

To build the project, easiest is with the vscode built in build function for ESP-IDF.

- Press F1 and search for build project.
- Press F1 and search for flash to UART.
- Press F1 and search for monitor.

Issues:

- Code is generally larger as there are fewer wrappers
- WebSerial is not installed
- Harder to debug
- Slow build
- Can be hard to install the framework
- If the camera init fails decrease log level to info, and increase ram to 4mb

## Tasks and issues

SPI communication is currently not working in both frameworks. The communication should be fixed or UART should be used. Documentation for the SPI communication is linked with description of frameworks.

## NUCLEO

The Nucle is responsible for storing sensor inputs, requesting image from the ESP, running the neural network for decision making and spinning the servos. Code for the nucleo is well constructed, and mostly autogenerated by the Cube IDE.

## STM32 Cube IDE

This framework suits the board well. When launching the .ioc project file many configuration can be applied visually.

Look for definitions and constant values in the header files. There is a pre implemented line following algorithm which can be found in the main.c file.

## Issues

- The project might not detect the X-CUBE dependency. This has to be linked manually in the ide by opening the .ioc. Then *Pinout & Configuration > SoftwarePacks > X-CUBE-AI > simplenn* and link the onnx file that can be found in the Nucleo main project root.
- C-CUBE-AI generates flags for the build that cause some build errors, such as no **heap\_malloc not found**. To resolve flags have to be deleted in project *Properties > C/C++ Builds > Settings > MCU GCC Linker > Miscellaneous*. Remove everything, `-u_print_float` is optional. This usually happens after each code generation.
- At debug if there is no line under the car the debug will be stuck in the the lighth sensor task, best way to debug is to lift the car with a wallet, so it does not roll off from desk.

## Tasks

- Handle master input from ESP. Communications have example in network.c, should be put into a separate file.
- Find suitable location for ESP on the board, which faces the direction the vehicle is facing, and in general has an angle towards the ground.

# Python Server

---

The python server is the main hub for communication with the devices. It starts a socket and the ESP connects to it based on the configured parameters on the controllers. The server waits for user input on the command line, and starts communication protocols based on them. If no user input is entered the server send a NO\_NEW\_COMMAND signal to the ESP, which continues as if nothing has happened. The server also has persistent data storage. Information that is inserted into the **ProcessedState** class can be saved into a json based db.

## Development

Server can be started with the *StartSocket.py* script. Current event handlers are implemented in the **Socket** class. The neural network is generated in the *ExportedModel* folder. The .onnx file generated must match the file in the Nucleo project root.

## Tasks

- When communication between the two controllers are established, data reseving needs to be implemented based on the incoming data structure.

## Simulaton

---

In the first semester of the porject a simulation enicorment was created in Carla to pretrain the network. Network performed well in the simulation, but we couldn't try them on the vehicle as it was misconfigured. In the future training in Carla could provide the starting weigths for the network.

## Contact

---

For more information or issues no contained in this project write to the following people:

Dániel Turóczy: Teams or e-mail: [turoczyd6@gmail.com](mailto:turoczyd6@gmail.com)