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, descrive the project for future develpopment.

Sections:

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

Dictionary

Reward - Reward is calcualted based on the network's decesion State - State of the vehicle (postion, speed, sensor data, image, decesion of network, reward)

Architeture

The system consisit of three main componenets:

- Nucleo Controller
- ESP32-CAM Controller
- Python Server

The Nucleo board is responsible for handling sensor infomration, 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 responisble 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.

Microcontollers

ESP

ESP is the commmunication master. All data flows though here between the server and Nucleo. There are currently two versions fo the code:

- ESPRESS-IDF VS-CODE
- Arduino IDE

Both of them are buit similarly, but each of them was uncapable 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 the mto the server. Due to computational capacities the image is grayscale. The smallest avaialble configuration is 96X96. infomration about this we could not found on the gitub, but in the ESPRESS-IDF we get this size as a suggestion. More infroamtion about camera configuraton can be found on the ESP32-CAM github. Definitons and values can be found in the header file.

We have sed 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 Nucle was a master controller for signalling that the ESP is ready for communication.

Development

Currently there are two alternatives to beign 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.

Adruino IDE

Code can be found here.

The main difference is that this code uses a serial monitos, accasable through browser. This was helpful in configuation as when the ESP is connected to the Nucleo there is no acces to the serial manually. Otherwise pacakges can be installed on the Arduino faster, and there are some convinient wrappers.

Issues:

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

ESPRESS-IDF

Code can be found here.

This framewrok provides robust development. It contains lots of examples and is well documented(link).

Issues:

- Code is generally larger as there are fewer wrappers
- WebSerial is not installed
- Harder to debug
- Slow build
- Can be hard to install

Tasks and issues

SPI comminication is currently not wokring in both frameworks. The communication should be fixed or UART should be used. Doucmentation 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 entwork for decesion making and spinning the servos. Code for the nucleo is well constructed, and msotly autogenerated by the Cube IDE.

STM32 Cube IDE

This framwework suits the board well. When laucning the .ioc project file many configuration can be applied viusally.

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

Issues

- The project might not detect the X-CUBE dependency. This hast 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 usally happends after each code generation.
- At debuf if there is no line under the car the debug will be stuck in the the ligth sensor task, best wa to debug is to lift the car wit h a wallet, so it does not roll off from desk.

Tasks

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