GyroCycle – Self-balancing bicycle

CS-358 MAKING INTELLIGENT THINGS – TEAM PROJECT PROPOSAL

By Tom Vrakking, Ondrej Zedka, Rémy Berguerand, Maël Imhof and Alexandre Hetet

Description

Main idea. Create a small scale bicycle that balances itself automatically using physics laws. The bicycle could stay balanced while idle (sationnary) and while moving (forward, backward, turning left or right).
Furthermore, the bicycle could be controlled through a software interface such as a web or mobile app, to make the project more interactive.

Extended idea. Once the main idea is implemented, computer vision could be introduced for the bicycle to drive itself given some instructions or goal.

The motivation behind this project is to play with the laws of physics and build something cool and fun that can be used as a demo for future EPFL students, physics students, …

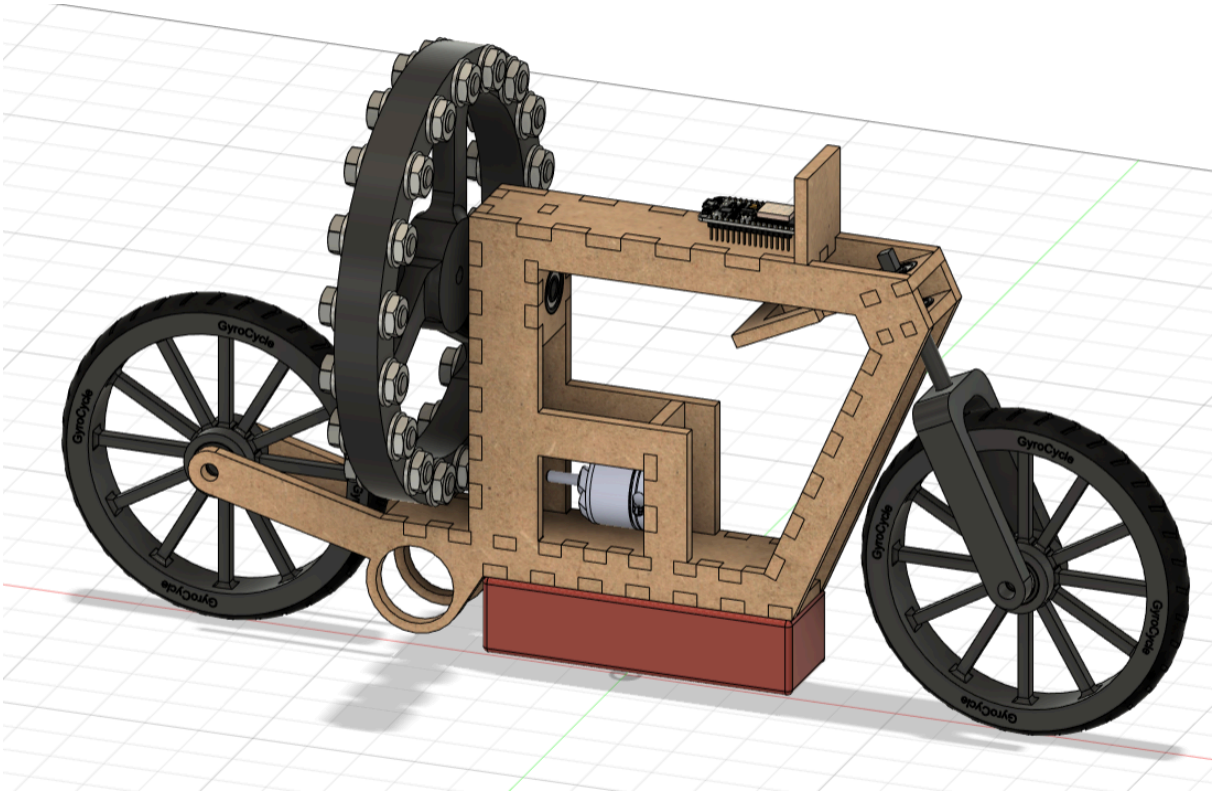
Milestone 1 - Gyroscopic Balancing

The first stage focuses on the ability for the bike to balance on its own while stationary or while moving/turning. The stage is divided in three sub-stages

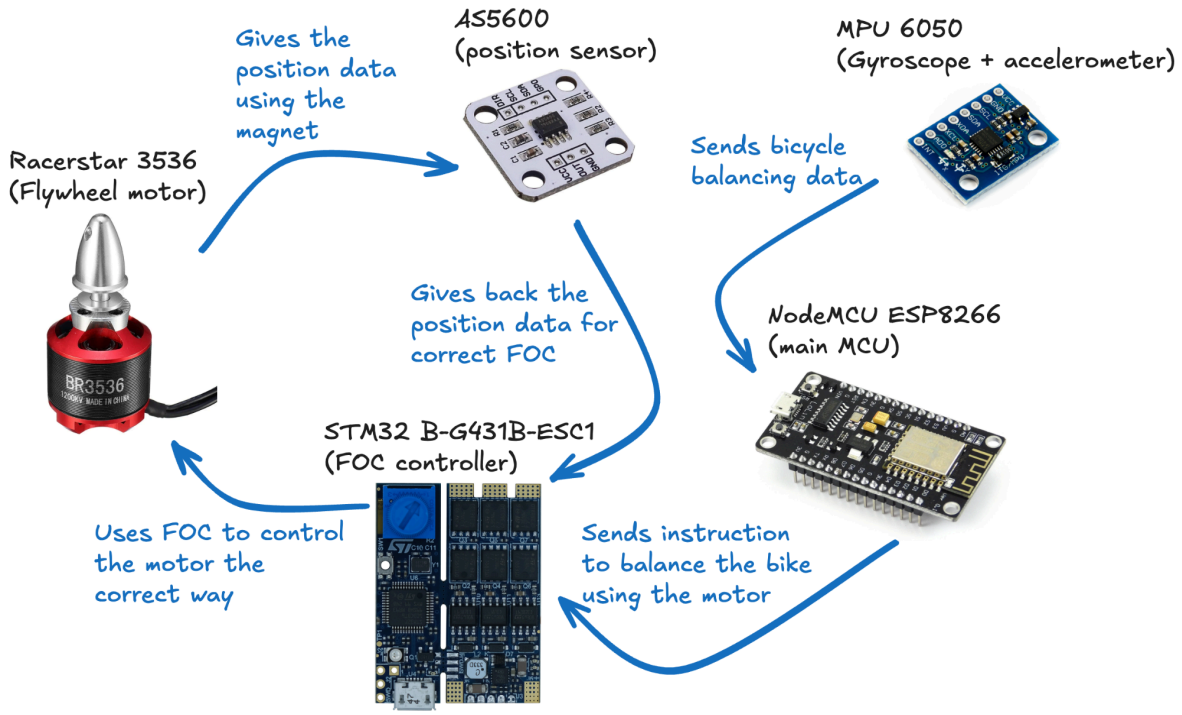
1. Balance when stationary
2. Make the bicycle able to move (forward, backward, turn left and right)
3. Balance while moving

Stationary balancing

The team will first focus on building the bicycle itself and making sure it is able to automatically balance while stationary.
An **accelerometer** and **gyroscope** will be used to measure the bike's tilt. The microcontroller will then adjust by accelerating the flywheel in the right direction to correct any imbalances.
Field-Oriented Control (FOC) will be used to control the wheel's motor with the required precision. In addition to the main microcontroller, a FOC driver will thus be needed.
At this stage, the bicycle should already be designed in such a way that adding components for computer vision later does not require a complete redesign.



Mechanical build for stationary balancing. Note that motors for propulsion or steering are not in there yet.



Architecture of the different electronic parts

Hardware

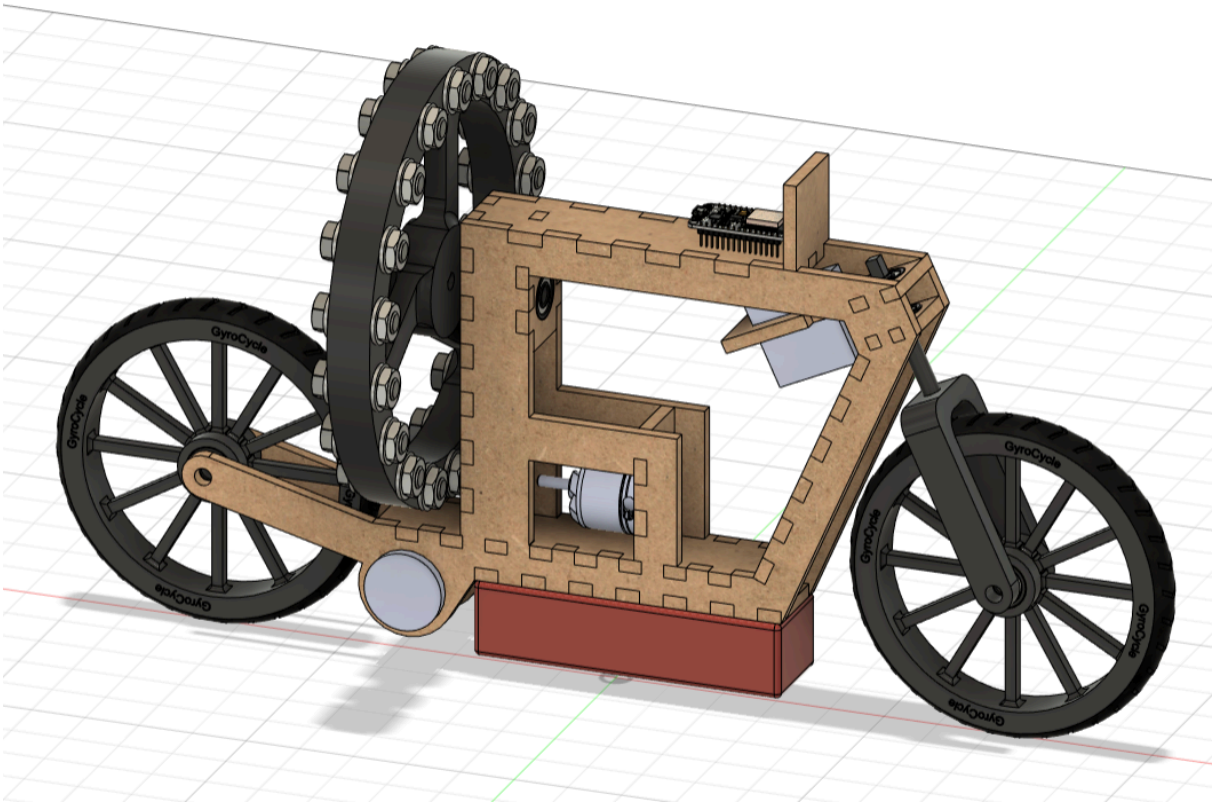
- Design and build the bike with computer vision in mind already
- Assemble and mount the electronics
- Make the bike look as pretty as possible

Software

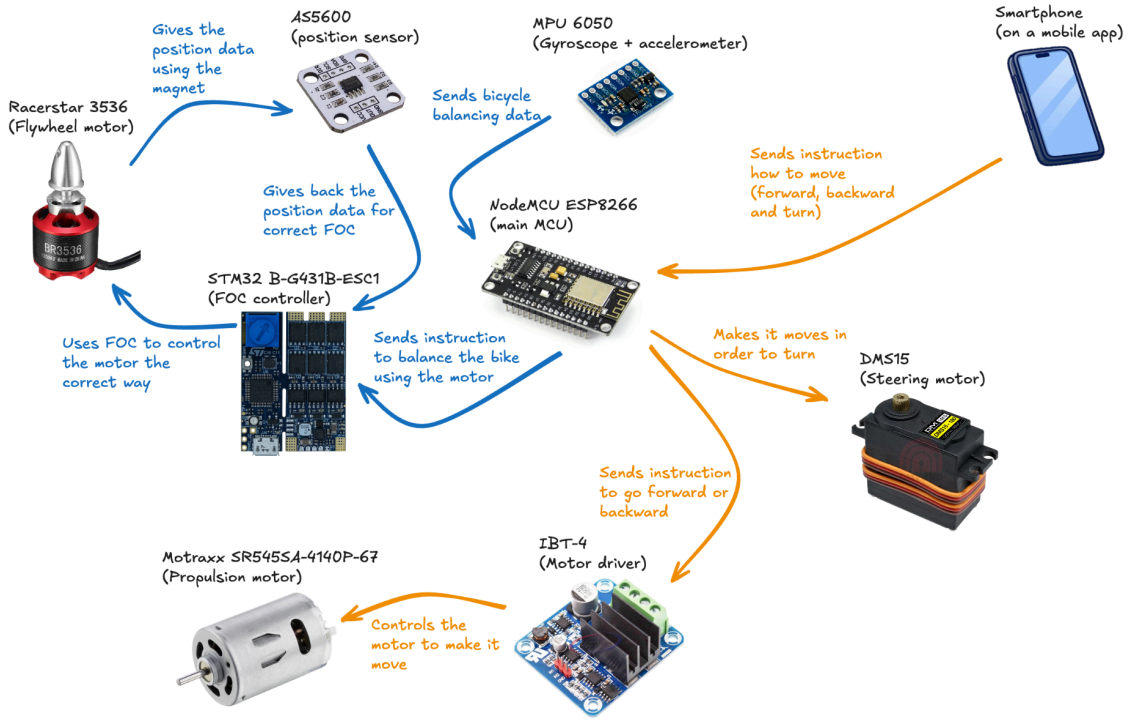
- Implement stationary balancing

Moving bicycle

Once the bicycle is able to stay balanced while stationary, the team will move on to making it able to move while still balancing itself.
This will require an additional motor for propulsion along with a servo motor for turning left and right.
The software will need to be updated to account for possible movement of the bicycle and to know how to balance it while either stationary or moving.



Moving bike. A propulsion motor is added at the bottom of the flying wheel and a steering servo is added on the front wheel's axis.



Architecture of the different electronic parts

Hardware

- Mount the propulsion motor and the servo

- Mount and connect the WiFi chip
- Mount and connect the camera
- Make sure the wheels can rotate smoothly

Software

- Implement a web portal or mobile app to remote control the bicycle
- Implement client-side software to receive and execute remote instructions

Moving balanced bicycle

The bicycle is balanced at rest and can move as well, it should also be able to automatically balance while moving. This will be the next and final step of the first milestone for this project.
Ideally, software adaptations should suffice to make the bicycle balanced while moving. If needed (if something goes wrong, and something will probably go wrong), the hardware will need to be adapted as well.

Hardware

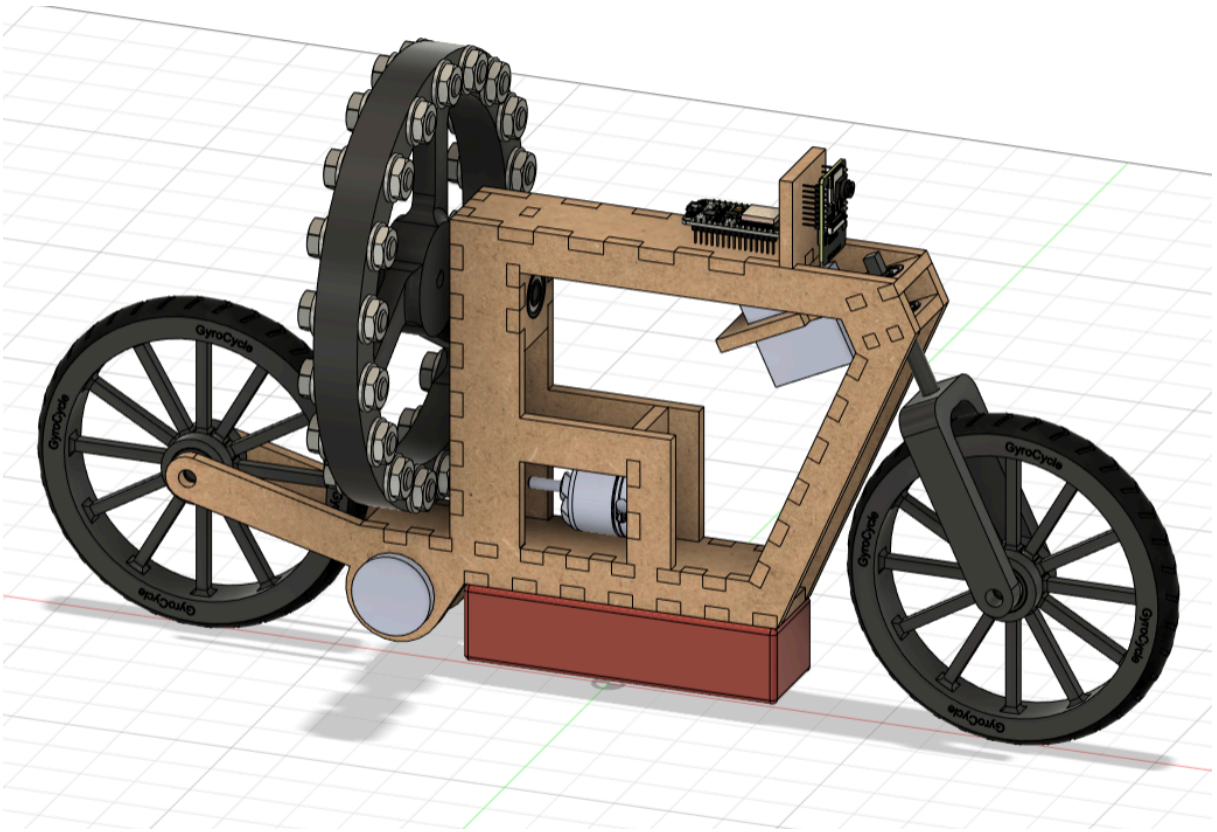
- Pray that nothing has to change
- Adapt the hardware if the software does not suffice

Software

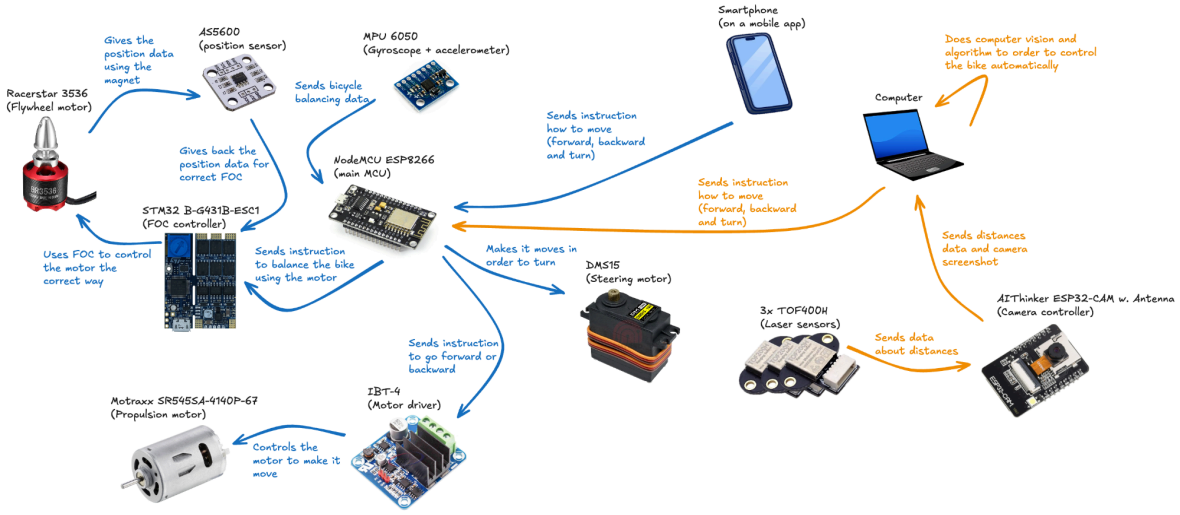
- Implement balancing for the moving bicycle

Milestone 2 - Computer Vision

The second stage of the project introduces computer vision for the bicycle to move automatically while avoiding obstacles.
If a camera was introduced on the hardware in milestone 1, it can be used, otherwise a camera will need to be mounted onto the existing assembly. Apart from this, changes to the physical assembly shouldn’t be necessary. The computer vision program should run on a laptop connected via WiFi and send commands just like the remote controller in milestone 1.
This milestone will hence be mostly software, since the main challenge will be to decide, based on a (relatively low-quality) video feed, which action to perform (turn left, turn right, …). The balancing will be handled by the bicycle itself and won’t require additional instructions from the laptop.



Addition of computer vision. Add an ESP32 camera at the front of the bike close to the other microcontroller.



Architecture of the different electronic parts

Hardware

- Mount and connect the camera if not done before

Software

- Implement a computer vision algorithm that, based on the image sent by the camera and the current state of the bicycle, sends commands to the bicycle to avoid obstacles

Optional improvements

Optionally, if there is time, the team will try to make computer vision work on the used microcontroller. As stated in the course manual :

Computer vision purely on an ESP32-CAM. It is possible, but so far in this course, teams have done the image processing on their laptops only.

Hence it would be an interesting challenge to tackle. It is however likely that time won't allow this optional part to be done.

Hardware

- Add cooling or additional connections as needed for running computer vision on the microcontroller

Software

- Minimize, simplify and optimize the software for it to run on a microcontroller

User Stories

The project represents an interesting demonstration of the laws of physics. and good entertainment for open doors at EPFL.

- As a **physics teacher**, I could use a self-balancing bicycle to demonstrate the laws of rotational physics to students in a very concrete way, as a project other students made.
- As an **EPFL open doors event organizer**, I could let people control the bicycle through its remote interface, to use it as an interactive demonstration of what is done at EPFL.
- As a **high-school student**, I would love seeing such a demonstration at EPFL open doors events, to show me how fun engineering can be and to motivate me to come to EPFL.

Known and useful resources

A self-balancing bicycle, though an interesting project, is not a brand new idea. There are several resources out there that can help build such an idea.

A particularly helpful resource will be the YouTube channel of ReM-RC, dedicated to balancing things. Specifically, three videos are very similar to the present team project proposal:

- Self balancing bicycle (reaction wheel bike)
- DIY 3D printed self-balancing bike
- Self balancing bike

Those projects are open-source and available on GitHub. They can already help in design and components decisions, and will later help implement software for balancing.

We will build a self-balancing bicycle using a flywheel and FOC to control said flywheel. An important resource for the project will be SimpleFOC, in particular their library and documentation.

At one point in the project, we will want the bicycle to move and to remotely control it from a mobile application. RemoteXY can help by making it easy to implement such an app and connecting it to the hardware easily without reinventing the wheel (pun intended).

Bill of materials

Category	Item (link)	Qty	Total Price (CHF)
MCU	**NodeMCU ESP8266 V0.9**[1] - Main microcontroller	**1**	3.-
Power	**3S-11.1V, 5500 mAh, 20C (110A)**[1] - LIPO battery	**1**	52.-
	Lipo protection board[1] - LIPO protection circuit	**1**	2.-
	XT90 male + female - LIPO connectors	**2**	13.80
	Joy-it SBC-Buck01 - Buck converter	**1**	6.80
Balancing	**MPU 6050**[1] - Gyroscope + Accelerometer	**1**	5.-
	Racerstar 3536[1] - Flywheel motor	**1**	15.-
	AS5600[1] - Flywheel motor position sensor	**1**	3.-
	STM32 B-G431B-ESC1[1] - FOC controller	**1**	20.-
Moving	Motraxx SR545SA-4140P-67 - Propulsion motor	**1**	11.95
	IBT-4[1] - Propulsion motor driver	**1**	6.-
	DMS15[1] - Servo motor	**1**	5.-
Computer vision	**AIThinker ESP32-CAM w. Antenna**[1] - Camera controller	**1**	13.-
	FTDI USB-to-serial - Adapter for ESP32-CAM	**1**	2.-
	TOF400H[1] - Laser distance sensors	**3**	18.-
Miscellaneous	**USB isolator**[1]	**1**	15.-
	Steel shaft 8mm x 500mm - Shaft for flywheel	**1**	3.35
	Pack of 8 Bones Bearing Reds - ball bearings	**1**	25.90
	GT2 Pulley 40 teeth 8mm Bore - For flywheel & wheel	**2**	11.90
	GT2 Pulley 20 Teeth 6.35mm Bore - For flywheel motor	**1**	4.20
	GT2 Pulley 20 Teeth 5mm Bore - For prop. motor	**1**	3.-
	Shaft adapter 3.2mm to 5mm - For prop.motor/pulley	**1**	5.50
	1 meter of GT2 belt 6mm wide - Belts	**1**	3.80
			249.20

1. Parts in stock for CS358 (said in manual)

Risk assessment, potential challenges

Such a project comes with challenges that will need to be addressed along the way. To prevent those challenges from impeding the success of the project, we list the ones we can think about below to plan in consequence.

- **Working with LIPO batteries**
  The team has no experience with working with LIPO batteries. Those components bring additional security risks to the project, along with somewhat more complex powering of the components.
  To mitigate risks relative to LIPO and to avoid damaging the circuit, rendering the project unusable, every team member is required to read the chapter 20 of the course manual and we will not hesitate to reach out to the course or DLL staff in case of any doubt or question.
  Also, we will connect all electronic components together (including the LIPO) without mounting them on the bicycle first, to make sure the wiring is correct.
- **Different power requirements**
  The project involves powering multiple sensors, actuators and microcontrollers that all have different power requirements. Managing power distribution and ensuring consistent supply without damaging components is crucial.
  We will build an early prototype of the power system to double-check all components are powered correctly and function as expected, without mounting them on the bicycle yet.
- **WiFi network stability**
  Giving remote instructions relies on using the WiFi of the ESP32 microcontroller to communicate between a phone and the bicycle. WiFi can be unstable or slow when used to give real-time instructions. This could affect the reactivity of the bike to user input, and later on to computer vision commands.
- **Robustness**
  The bicycle will likely fall more than once when we test the balancing software. To prevent it from getting damaged, a team member should always be ready to manually catch the bike before it hits the ground.
- **Computer vision and real-time video data**
  The second milestone introduces computer vision, which implies sending real-time video data from a microcontroller to an external laptop via WiFi, processing the real-time video data on the laptop as quickly as possible, and sending back commands to react to the situation.
  The team will start by testing using only the ESP32 microcontroller and some simple obstacles to learn how to implement an algorithm that works on a low quality video flow.

Organization

Milestone 1 (MS1) - self-balancing bicycle that can stand still or move, controlled through a web-app/mobile app

Idle but balanced Milestone B	Construct a downscaled bicycle that balances itself at rest
B1.1 **Mechanical**	● Design and build a self-balancing bicycle with movement in mind already ● Document the hardware development in detail
B1.2 **Balance software**	● Implement software to keep the bicycle balanced when idle ● Test the software and check the balance
Moving and balanced Milestone M	Add the capacity to move the bicycle through a web/mobile interface
M1.1 **Adapted design**	● Add missing hardware for movement (motors for the wheels, motor for turning left and right) ● Document the hardware development in detail
M1.2 **Control app**	● Implement a web/mobile app to control the bicycle through a WiFi connection.
M1.3 **Controllable bicycle**	● Implement code for the microcontroller that will receive WiFi instructions from the app ● Marry the app and the hardware ● Adapt the balancing software to account for movement

Milestone 2 (MS2) - integrate computer vision so that the bicycle can automatically move and avoid obstacles

External computer vision Milestone E	Add a camera to the bicycle along with real-time processing with computer vision algorithms for the bicycle to drive itself
E1.1 **Adapted design**	● Add a camera to the bicycle both in the design and on the actual physical design ● Document the hardware development in detail
E1.2 **Computer vision software**	● Implement software to process the feed of the camera and output instructions to the bicycle
E1.3 **Self-driving bicycle**	● Marry the adapted design with the camera and the computer vision software ● The computer vision software should run on a separate laptop for now
Mounted computer vision Milestone O	O for Optional. If there is time, adapt the computer vision software so that it can run directly on the microcontroller
O1.1 **Adapted software**	● Adapt the computer vision software (scale it down, optimize it) so that it can run on the microcontroller

Detailed Plan

Wk	Alexandre	Maël	Ondrej	Rémy	Tom
7	Laser-cut and assemble MDF parts	3D-print and assemble the flywheel (with screws and all), research how to do balancing software	3D-print the front wheel's holder and the wheels	*TBD*	Electronics prototype (connect electronic components), implement control of the motors and sensors by the microcontroller
8	Mount microcontrollers, drivers, LIPO and LIPO protection circuit	Implement balancing software for idle bike	Mount the flywheel, its motor and the pullies/belt on the MDF frame	Solder and wire the electronic circuits. Avoid wires getting stuck in wheels.	Test propulsion and steering without balancing
9	Design a protocol to transmit commands to the bike, Implement a web app to transmit commands to the bike (RemoteXY?)	Test idle balancing software and make it work	Research what software to change to balance while moving	Design a protocol to transmit commands to the bike, Implement a web app to transmit commands to the bike (RemoteXY?)	Implement reception of instructions from outside (via Wi-Fi) on the microcontroller
10	Research computer vision algorithms with ESP32 image and laser sensors	Research computer vision algorithms with ESP32 image and laser sensors	Implement and test balancing software for moving bike	Implement control of ESP32 and laser sensors, brainstorm computer vision algorithms	Test remote control using the web app
11	*TBD*	*TBD*	Implement a computer vision algorithm	Implement a computer vision algorithm	*TBD*
12	*TBD*	*TBD*	Implement a	Implement a	*TBD*

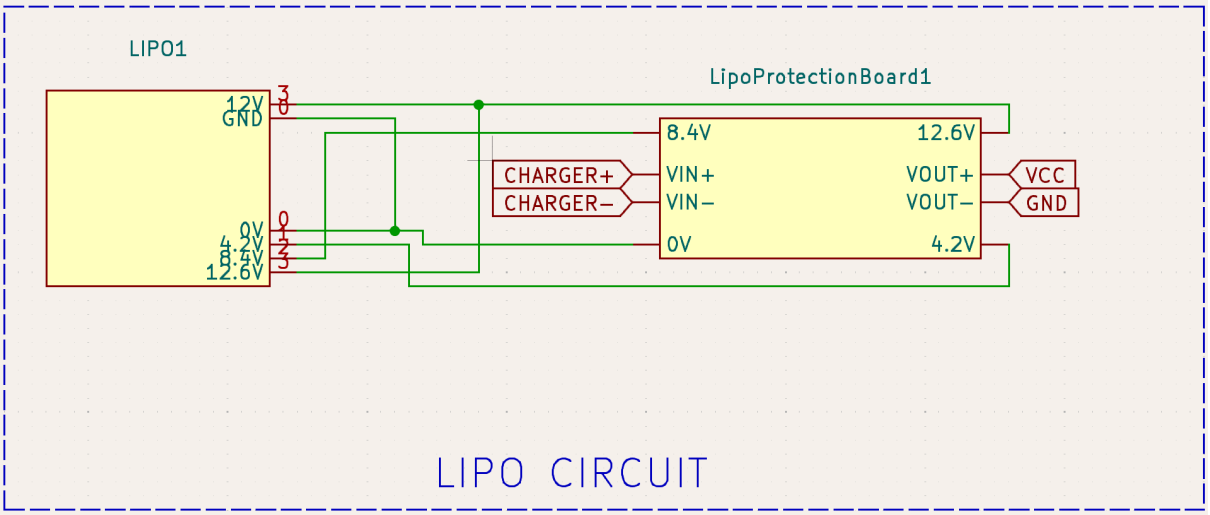
			computer vision algorithm	computer vision algorithm	
End of Milestone 1					
13	Test and adjust computer vision model	Test and adjust computer vision model	Test and adjust computer vision model	Test and adjust computer vision model	Test and adjust computer vision model
14	*TBD*	*TBD*	*TBD*	*TBD*	*TBD*
End of Milestone 2					

Planning ahead is a difficult task, especially for projects in domains you do not know that well. The above is an optimistic plan that tries to account for problems by introducing *TBD* slots where the people are not assigned to anything in particular for now.
The tasks will be assigned during the SCRUM meetings anyway, this planning is only a tentative list of steps to reach the objective of the project.
Also, the planning highly depends on when the needed material will be delivered by suppliers. If hardware comes earlier or later than anticipated, the plan could drastically change.

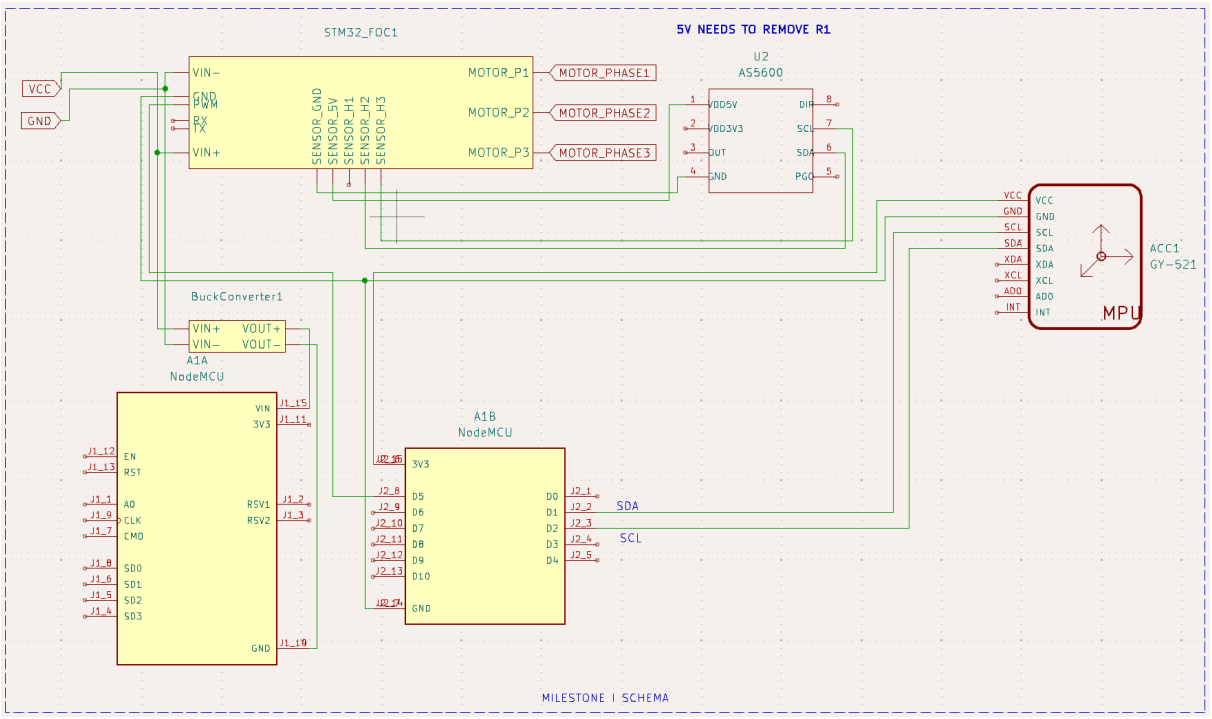
Appendix

Electrical schemas

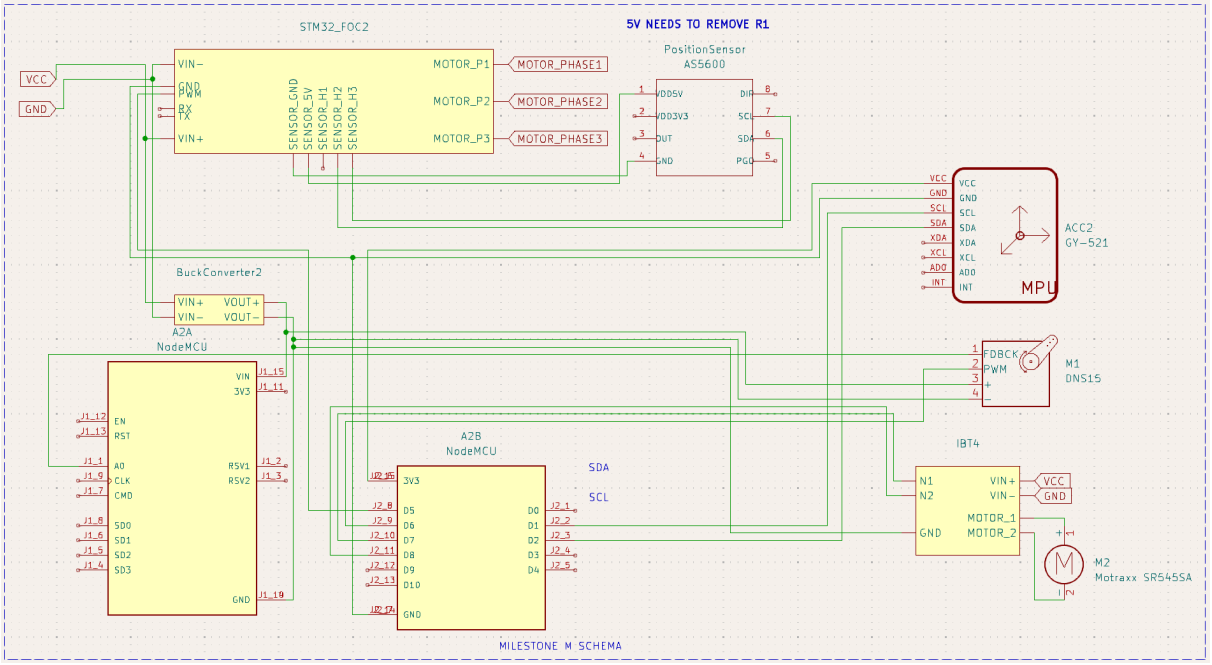
Note : VCC and GND are labels, and are connected to one another (i.e. all VCC tags are linked together with a physical wire, using labels is just for clarity)



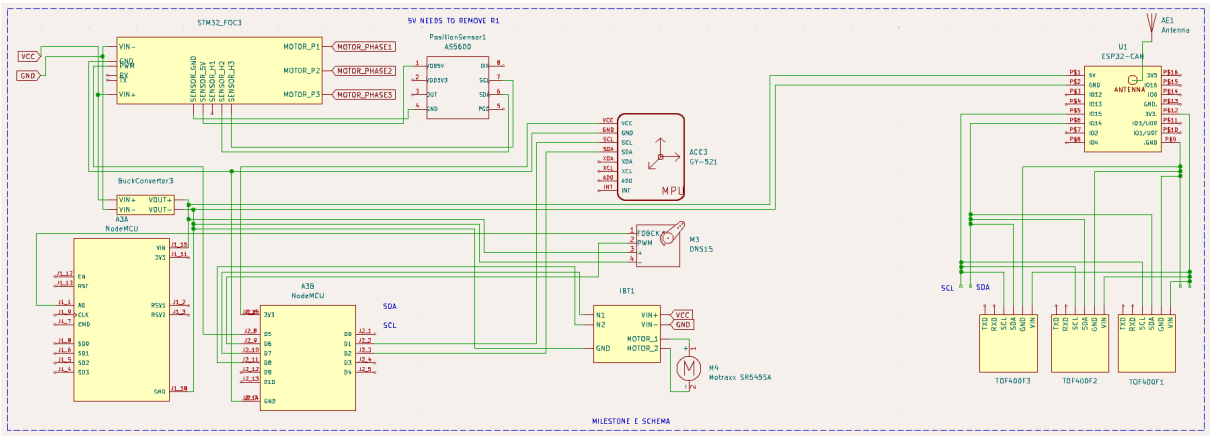
Schemas of the wiring of the LIPO battery to the LIPO Protection board



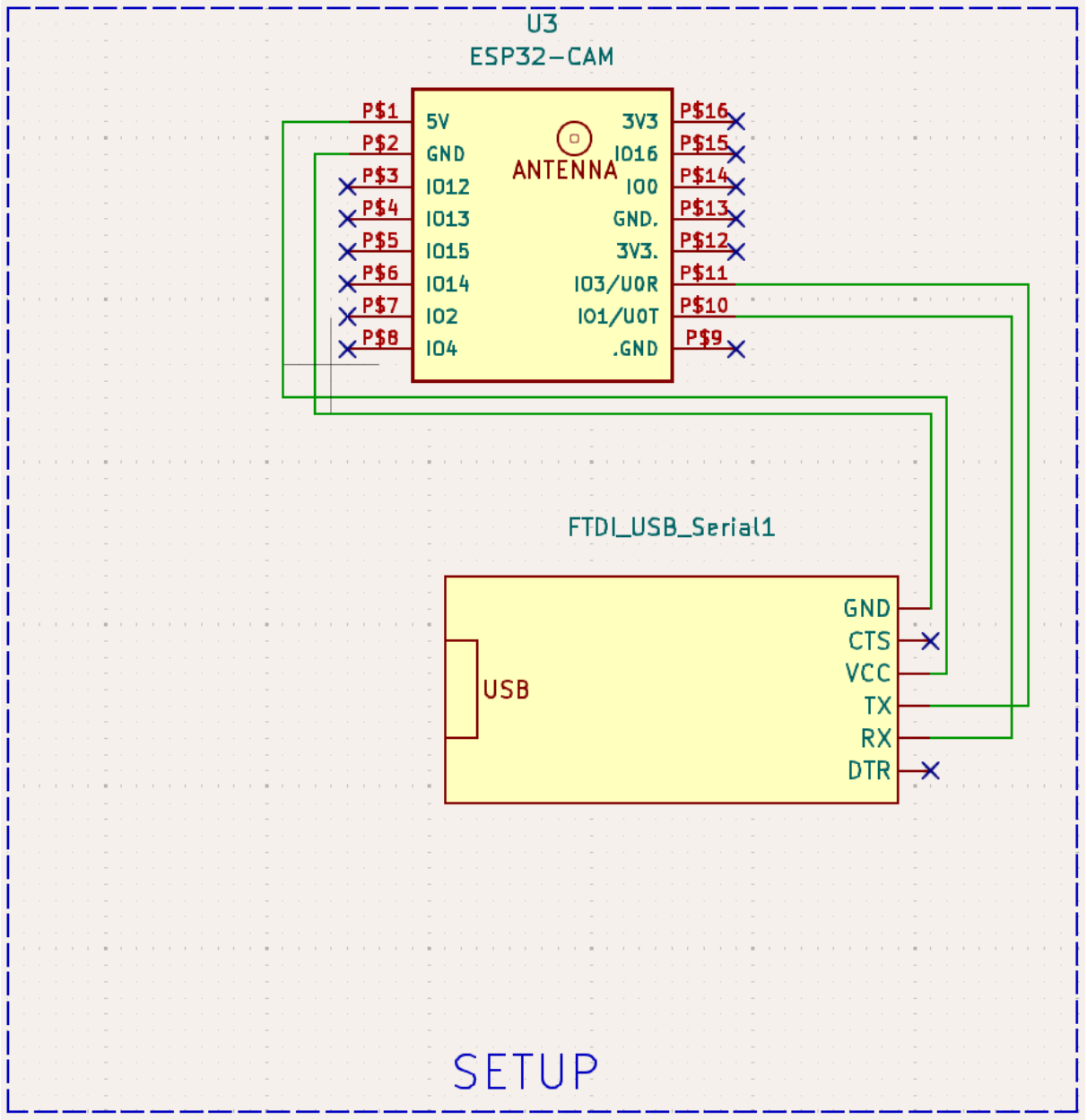
Schema of the Milestone I (i) wiring



Schema of the Milestone M wiring

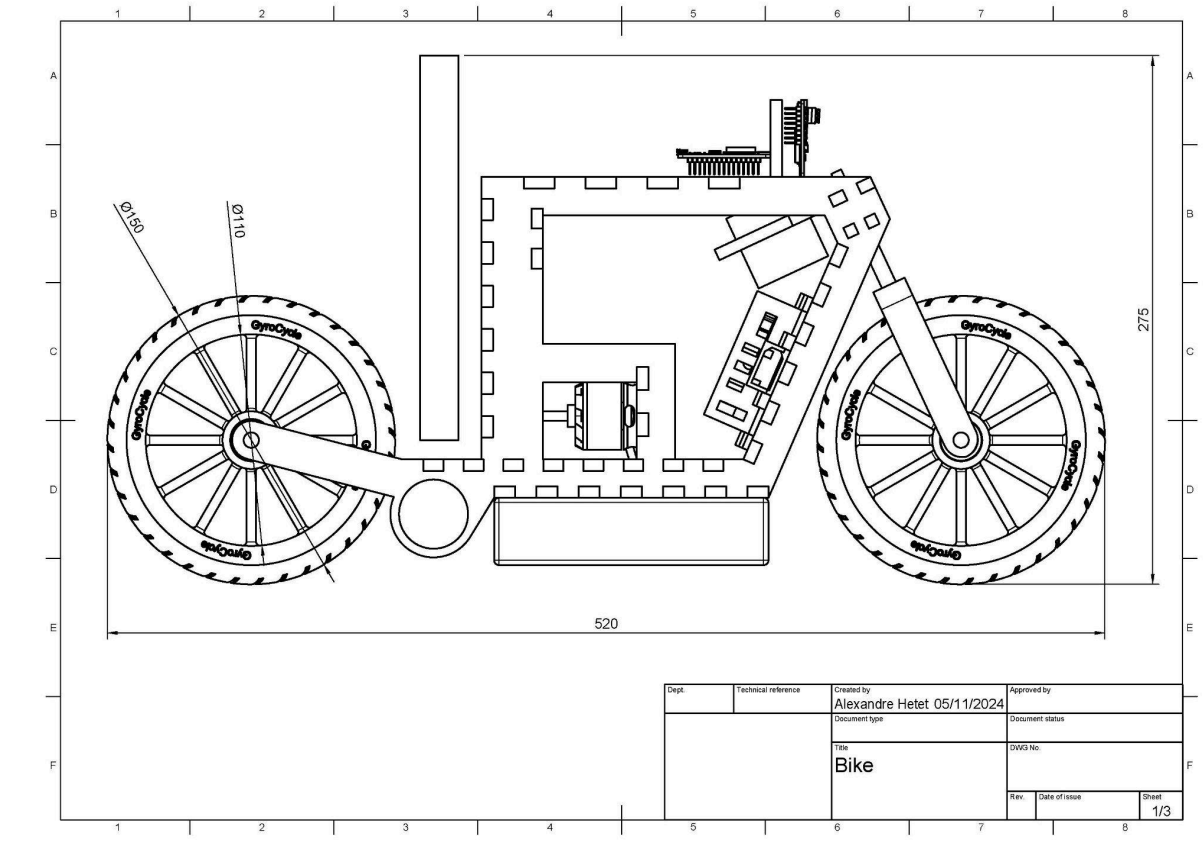


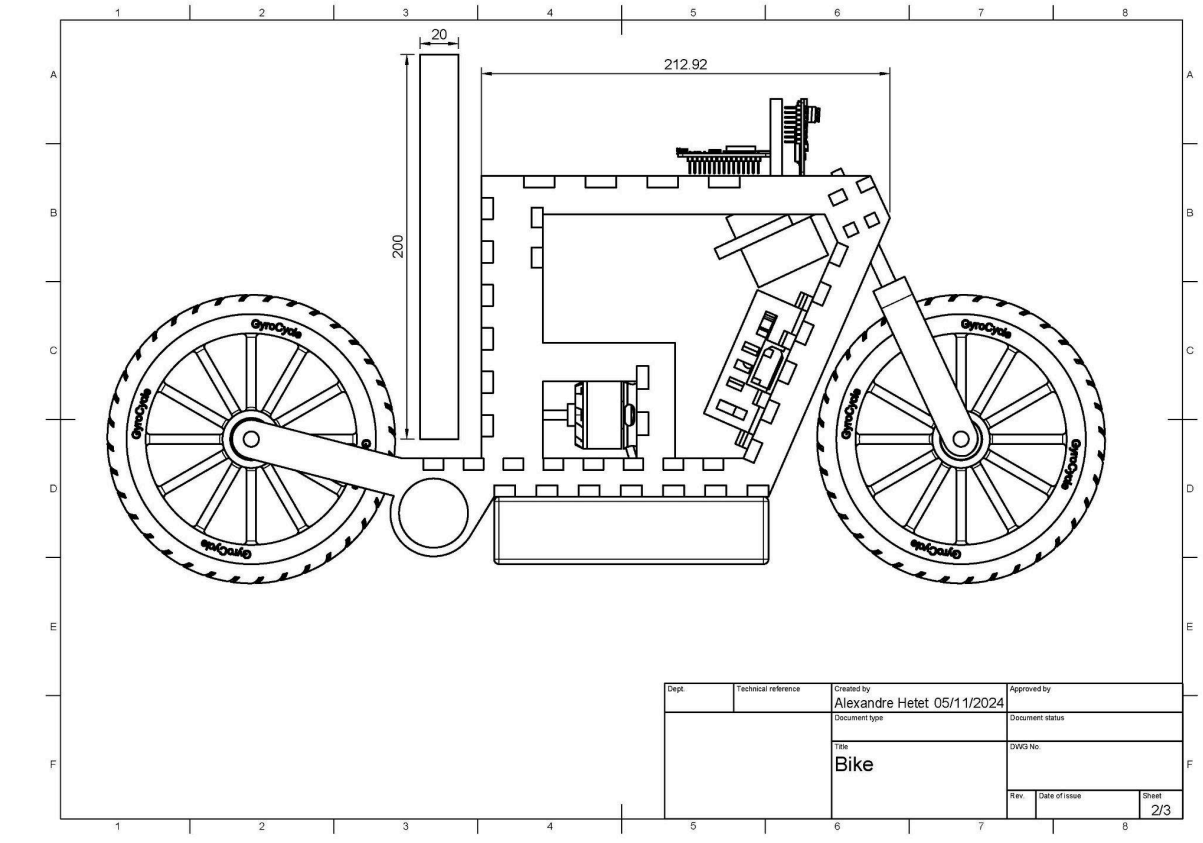
Schema of the Milestone E wiring

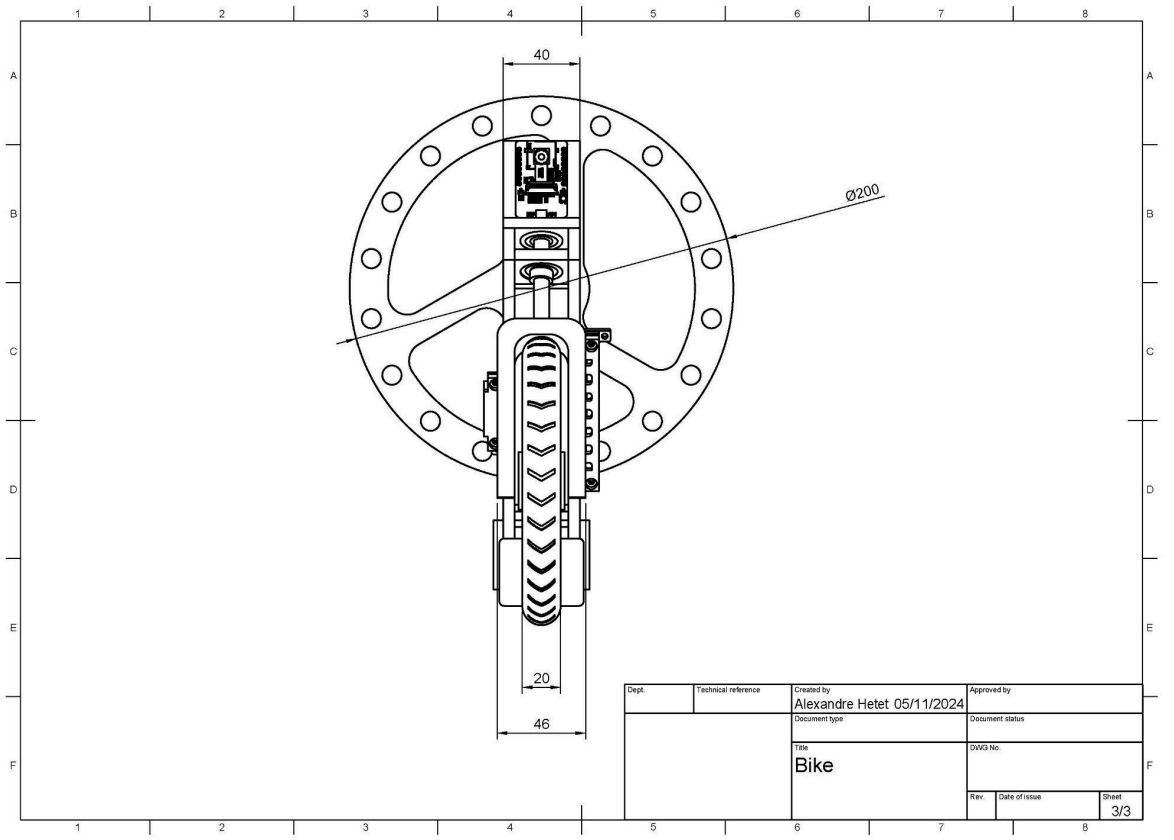


Schema of the setup wiring necessary for first initialization of ESP32-CAM

Global view of the size of the bike






40
Ø200
20
46
Dept.
Technical reference
Created by
Alexandre Hetet 05/11/2024
Approved by
Document type
Document status
Title
Bike
DWG No.
Rev.
Date of issue
Sheet
3/3