Proposal: BeetleBot

Anna Barberis, Omayma Hadiji, Jonas Zimmermann,
Adèle Monjauze, Anas El Madhi

March 2025

1 Description

The idea behind BeetleBot: BeetleBot, otherwise known as a Hexapod, is a six-legged robot with 3 degrees of freedom on each leg. The idea is to mimic the walking pattern of an insect, or more specifically to implement a tripod-like motion. Walking on flat ground, a hexapod can implement many different gaits to move itself. The most popular ones include tripod gait, reverse, and crab walk-like motion. The Beetlebot will also be able to interact and adapt to its surrounding, narrowing its stance to go through an obstacle-ridden path or avoiding objects ahead, to the sides, or behind itself. Thus, a hexapod demonstrates flexibility in how it can move and balance its weight. Our robot will respond to an input to direct itself, through a remote connection, like a game controller. It will also be able to make decisions based on its environment.

A deeper understanding: Stability is one of the keys to building a Hexapod. Using inverse kinematics equations, we will play with dynamic and static stability. Just like a human being can stand on one leg and shift his center of gravity, the same can be done for an hexapod (i.e. stand on half of its legs). If the center of mass of the robot is not probably balanced, the Hexapod will tip over.

1.1 Hardware

The selection of our electrical components was largely influenced by the choice of servos. To stay within our allocated budget while taking on the challenge of using relatively weak servos, we opted for SG90 servos to move the legs. They are lightweight, cheap, demand low power, and we have seen an existing project online that seemed to work using the full 18 of them. To connect the 18 + 1 servos together, we included a PCA 9685 PWM multiplexer. We also included 6 different servos, more powerful, but heavier and pricier. Those were included to be used on the weakest joint of the legs, the one that would bear the most weight, and that would be the most susceptible to break our SG90 servos. We plan to try with the weaker

servos first, and then decide later on if we want to replace them.

One of the main idea of this project was to have a connection between the robot and a remote controller directing the robot. This is why we decided to use a WIFI-Bluetooth compatible board. What oriented our choice to the ESP32-Devkit V1 board is its small size, as we want to avoid weighting down our robot as much as possible, and also that it had a large number of pins, which could be programmed to fit different protocols or interfaces like PWM (used by servos).

To enable the robot to interact with its surroundings and adapt its walking patterns based on environmental awareness, we selected 4 laser distance sensors. We chose laser sensors over alternatives due to their lightweight design, compact size, and suitability for our needs. Although integrating them may present some challenges in terms of coding, they are ideal for this application. To get a full view of the perimeter of the beetlebot, we will mount 2 sensors on one additional servo (the 19th one, an SG90), back to back. To cover any existing blind spots, we will add one sensor on each side of the mounted servo.
Since these sensors require an I2C interface, we needed an I2C multiplexer, that provided the sufficient number of pins with a unique address, to allow the microcontroller to differentiate them. The TCA9548A I2C multiplexer offers a unique I2C bus for each sensor.

To power our entire system, we chose a 7.4V 1000mAh 30C LiPo battery. This battery will supply power to the servos, microcontroller, and sensors through a DC buck converter that steps down the voltage to 5V. We aim for the robot to be self-sufficient with a single battery. As the servos each draw about 350mAh (stall) - 6.65A in total, the sensors about 40mAh - 160 mAh in total, and the microcontroller less than 100mAh, we estimated that one battery would be enough, given that all the servos won't realistically move at the same time.
However, in the event that power consumption exceeds our expectations, or if we decide to upgrade to larger servos (which each draw 1.5A at 5V), we have opted to purchase two identical batteries for added capacity and reliability.
To connect the batteries to the system, we will use a 2S battery protection circuit, which will allow for a current draw of up to 10A, something that should be more than enough. We will also implement a switch, rated for the appropriate voltage and current, providing a means to safely power off the system when needed. More details on each components can be found in this section.

Omitting cables, screws and whatnot, the overall weight of all of our components should range up to 365g.

1.2 Software

We intend on implementing different gait patterns (crab-walk, reverse, tripod, forward...), which will involve mainly mathematical computation, to decide on angles, and positioning of the servos. The core of the software will focus on coordinating the movements of all the

servos, deciding which one should move and how the rest should adjust to counterbalance the motion. This will be a critical aspect of the project's software development.

The next step software-wise to our project will be to code the laser distance sensors, and effectively let the objects around the robot change its way of moving. One of our key features will be enabling the robot to adopt a narrower stance to navigate between two obstacles, two walls. It will also be as simple as avoiding an object to any side of the robot, and pick a different path. This phase of the project will introduce more complexity and provide a more dynamic and interactive challenge for the robot's navigation system.
In addition, we plan to integrate a remote-controller as an input method to control the robot's directioning. The communication will go through the microcontroller, using WIFI or bluetooth. As an illustration, the idea would be that pressing one direction on the controller (a simple one composed of four arrows) would active certain servos and make the robot move in that requested direction. This software side could add user interaction to our beetlebot.

1.2.1 Inverse Kinematics (IK)

Inverse kinematics (IK) is the process of determining the joint (servo) positions required to reach a specific coordinate in space.

With our completed CAD design, we can accurately measure the lengths of each segment of the BeetleBot's legs, which is crucial for performing IK calculations. These calculations allow us to determine how each servo should move to achieve precise leg positioning.

For a more detailed explanation, refer to the IK documentation in our GitHub repository. In brief, implementing IK enables us to program different gaits (walking modes) and is fundamental to achieving smooth and controlled movement of the hexapod.

For a deeper dive into IK equations for hexapods, the following resource provides a comprehensive overview.

1.3 Structure

The skeleton of our BeetleBot will primarily be constructed through 3D printing. Given the need for intricate and flexible, robust parts, this approach is ideal for creating the best possible structure of the bot. The design features a central base that will house all the electrical components, serving as the "main body" of the bot. We want this center base to be as light, compact, and sturdy as possible. Its center of mass should be centered around the geometrical center of our beetle bot, to avoid tipping and ensure stability during movement.

The six legs will also be 3D printed. Each will consist of three main parts, most of which will be designed around the servos that drive them. By separating the legs into modular components, we can incorporate mobility at every joint where the parts connect, allowing for efficient movement through servo-actuated joints.
The first joint, located at the connection point with the central base, will act as the hip joint, enabling forward and backward movement of the leg. The second joint, located at the knee,

will allow up and down motion (knee folding movement). The third joint, located at the foot, will enable forward and backward movement, complementing the knee movement.
The leg dimension needs to be fine tuned to improve the weight distribution, and overall liberty of motion of the bot.
To improve the grip-ability of the legs, and avoid the robot from slipping, we may use rubber on the tips of the leg.
To protect our servos while trying out different designs and structure, we will consider adding ball casters extended by a printed piece so that they can support the center base. The idea is to use them strictly as 'training legs'.

2 Technical drawings

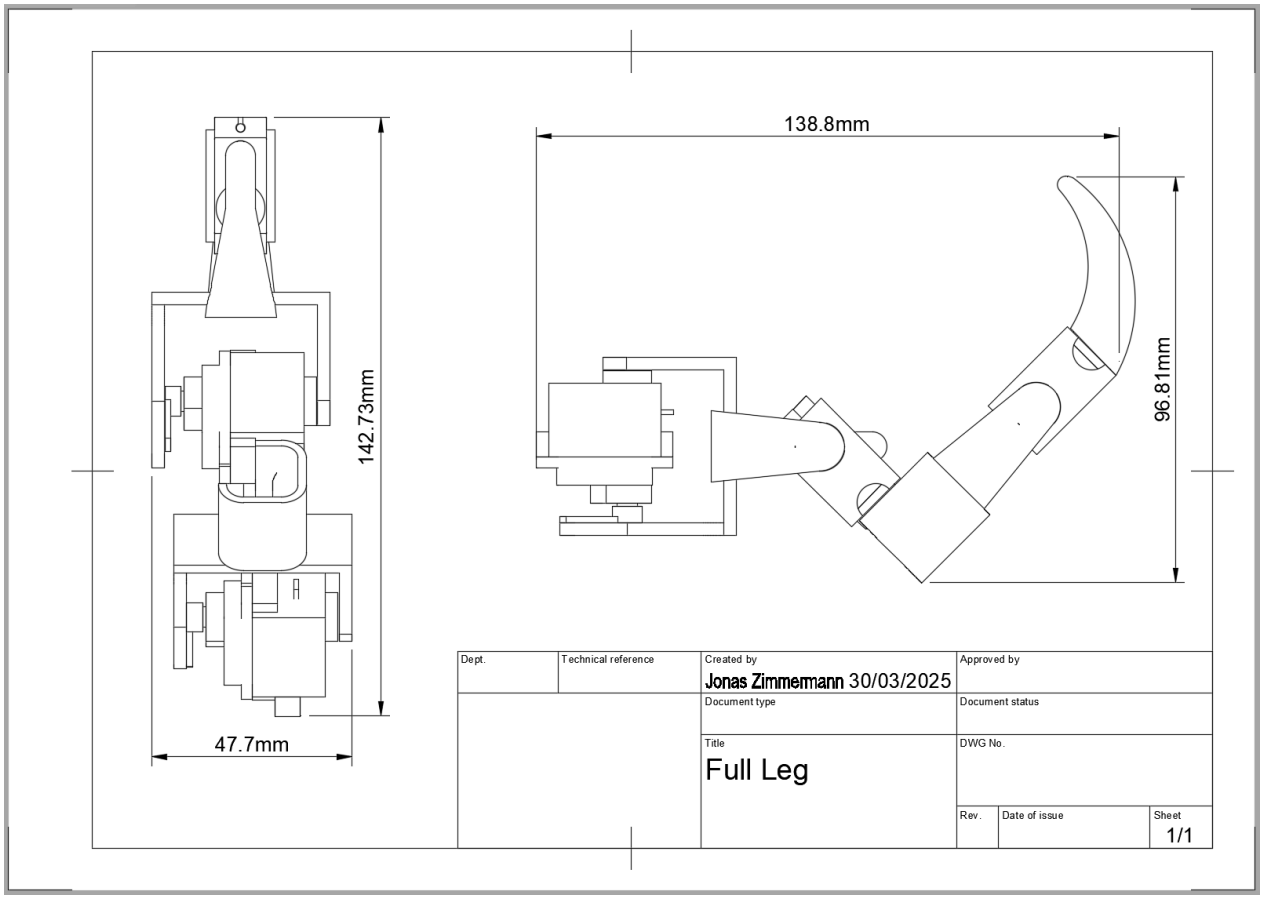

Figure 1: Drawing of a leg made with Fusion360

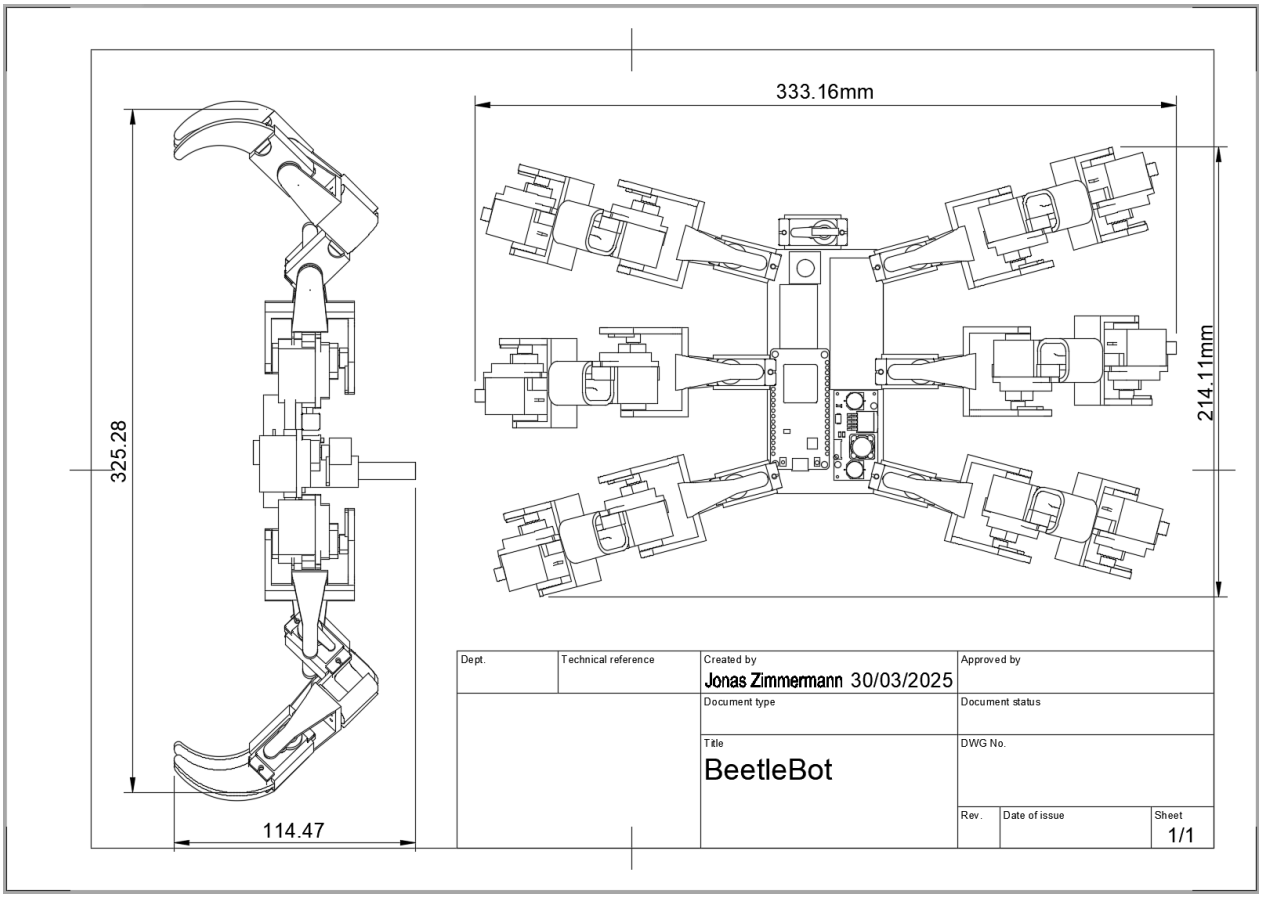

Figure 2: Drawing of the full bot

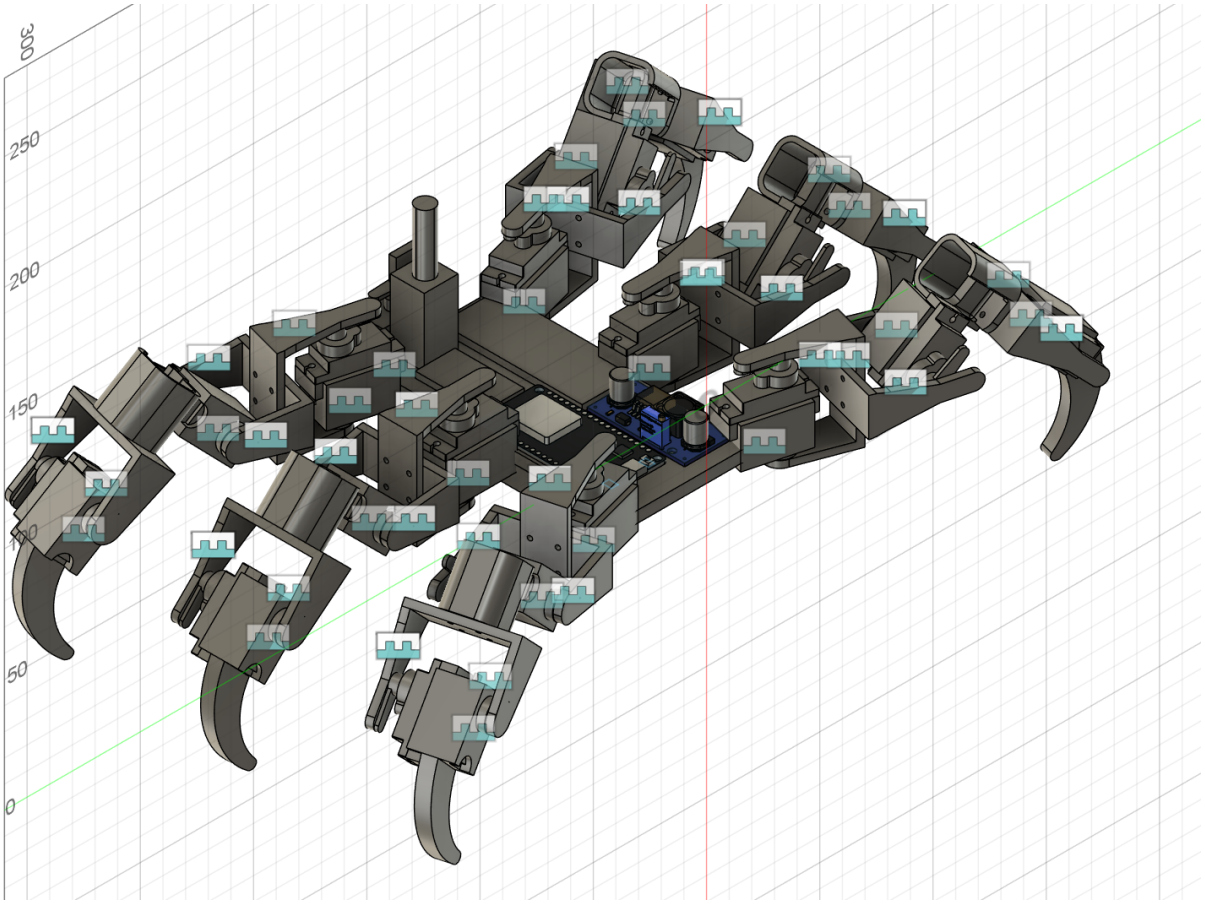

Figure 3: Screenshot of the bot made in Fusion360

2.1 Circuit diagram

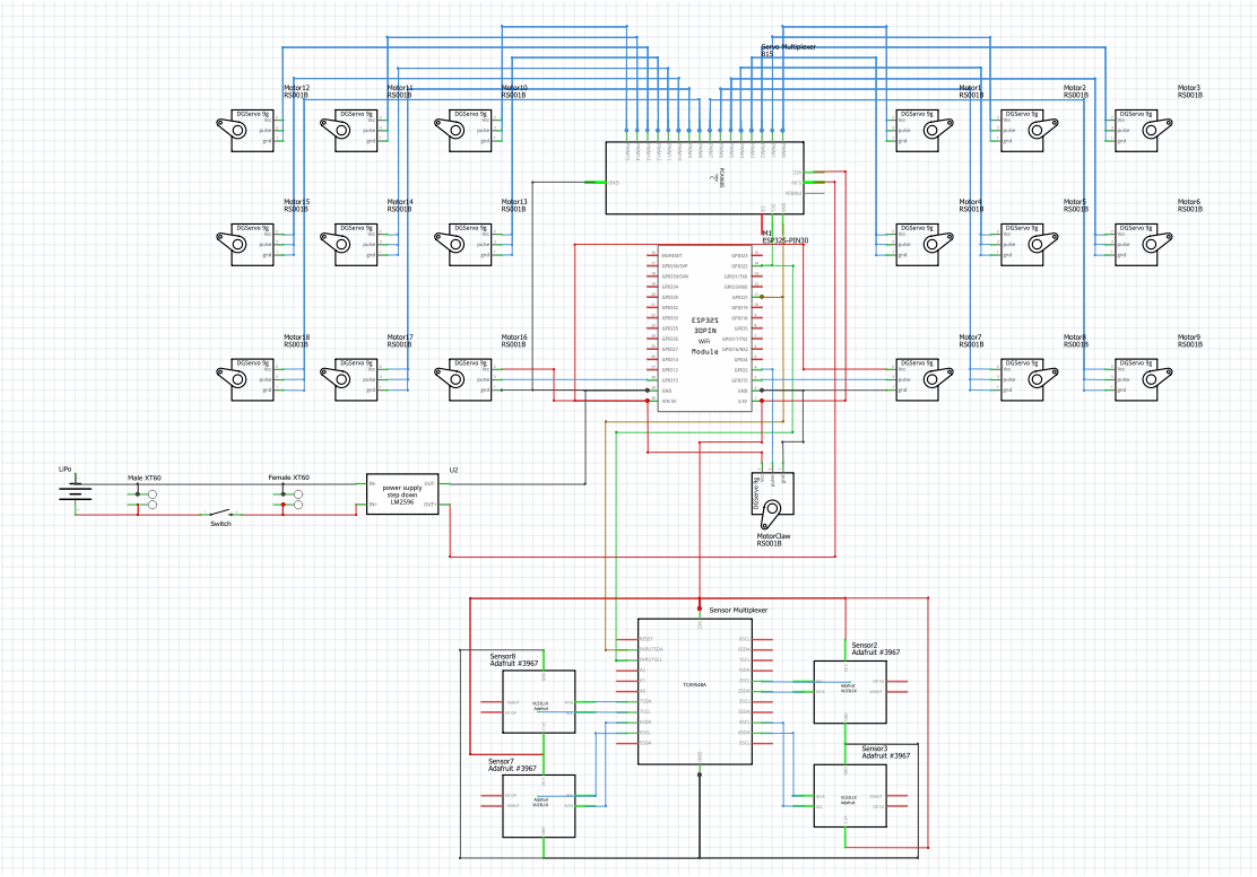

Figure 4: BeetleBot circuit diagram

3 Bill of Materials (BoM)

Category	Description	Quantity	Price
Power	LiPo battery 1000mAh 7.4V 30C XT60	2	17.5
Power	XT60 male connector	2	2.1
Power	XT60 female connector	2	2.7
Power	Charging cable XT60	1	5.6
Power	LiPo protection board for 2S	2	< 10
Power	LM2596 Buck Converter	1	6.5
Power	On/Off Switch	1	6.61
Wifi/ Bluetooth connection	Esp32 Devkit V1 Board	1	7.07
Moving	SG90 servo motor	19	57
Moving	MG996R servo motor	6	68.94
Moving	TOF400C laser sensor	4	20
Sensors	TCA9548A I2C multiplexer	1	6.14
Servos	PCA 9685 PWM multiplexer	1	12
Miscellaneous (as training legs)	Ball casters	4	9
		TOTAL	**231.16**

Note: ***for the protection boards, our current consumption should not exceed 10A***

4 Risk Assessment

4.1 Weight and Servo Strength

One of the main challenges we foresee is related to the weight of the hexapod and the strength of the servos. We are using relatively weak servos, so it is crucial that our design remains as light as possible. Achieving this will be challenging, as reducing weight often comes at the cost of structural integrity or functionality.

To mitigate this risk, we have brainstormed potential solutions in case the servos are unable to support the hexapod's weight:

- Incorporating ball casters under the base: This would allow the hexapod to roll on small metal balls rather than relying entirely on the servos to lift and move its body.
- Optimizing material choice and frame design: Using lightweight yet sturdy materials such as carbon fiber, aluminum, or 3D-printed lightweight plastic could significantly reduce weight without compromising strength.
- Reevaluating servo selection: If initial tests indicate that the servos struggle too much, we will need to consider stronger alternatives, though this would impact power consumption.

Before implementing any of these solutions, we plan to prototype the hexapod with the lightest and most efficient design possible. This will allow us to assess whether the additional support mechanisms, such as ball casters, are necessary.

4.2 Servo Movements and Walking Mechanics

While we have a general idea of the gait patterns our hexapod will use (e.g., tripod gait), we have yet to determine the exact servo angles required for each step. This is not a trivial issue, as each leg will be controlled by at least two servos (one at the hip and one at the knee), and their precise coordination is essential for stable movement.

Key challenges include:

- Angle calculation and optimization:
    - Each servo must move within its range while maintaining balance and fluidity in motion.
    - The stationary legs (those on the ground) will still need to move at the hip to follow the body's displacement.
    - Adjusting these angles manually could be a tedious process, requiring a mix of trial-and-error testing and kinematic calculations.
- Inverse kinematics (IK) and load distribution:
    - While we have done some preliminary research on inverse kinematics, and there is lengthy documentation available online about how the equations were determined for certain Hexapod designs, we still lack a deep understanding of how to minimize the force applied to the servos during movement.
    - The lever arm effect (longer leg segments increasing torque requirements) must be taken into account.
    - We can reference existing hexapod designs available online to refine our approach. (See Section 8 for such examples)

As mentioned earlier, if we cannot find an optimal leg design, we may still implement rolling support under the base to reduce strain on the servos.

4.3 LiPo Battery Use

Using a LiPo battery to power the entire system (servos, ESP32, sensors) adds complexity in terms of voltage regulation and current delivery. If the battery cannot supply enough current during peak loads (when multiple servos move simultaneously), it may cause voltage drops, system instability, or even resets.

Why it's a challenge:

- We're powering many components from one battery.
- Servos can pull a lot of current when moving together.
- If the voltage isn't stable, the microcontroller might crash or the servos might glitch.

How we'll deal with it:

- Use a LM2596 buck converter to get a steady 5V out of the LiPo.
- Test how much current the servos actually draw under load.
- Monitor the battery voltage so we know when it's getting low.
- Use the second battery to split the usage

4.4 Software Challenges

4.4.1 Control of the servos

Controlling 18 servos in software, especially in coordination, brings several challenges related to timing, synchronization, and responsiveness. We'll need to ensure that the walking behavior is smooth and that servo commands don't conflict or overload the control bus.
We can:

- Use non-blocking code and timers to ensure responsiveness.
- Start with basic movement primitives (e.g. lift leg, move forward) and build up to full gaits.
- Simulate and visualize movements if possible before testing on hardware.

4.4.2 Remote Controller

If we implement remote control (e.g. over Bluetooth or Wi-Fi), we introduce another layer of software complexity. Parsing incoming commands, dealing with connection issues, and integrating this with walking logic needs to be done carefully. Indeed, handling both remote control input and real-time walking requires multitasking, and there is a risk of tight coupling between communication and motion logic, which would make debugging harder.
For this issue we therefore can:

- Keep communication logic modular and separate from walking control
- Define a clear and minimal communication protocol (e.g. fixed-length messages or JSON)

5 Organisation

5.1 Milestones

To organize our work, we will divide the task into three main milestones.

First off, we will focus on building a simple version of our bot, carefully designing the 3D parts, as every design choice will impact the successful-ness of our robot. Given that the weight of the components and their placement will impact the stability of our BeetleBot. Once this is secured, we will implement a simple gait pattern to get familiar with the servos, such as the forward walk, without any added challenge. It will be mainly hardware focused, with an introduction to the software side at the very end of this milestone. This milestone is the most important one, as most of the challenges that this project arises will be tackled : weight management, physics, or simply communicating with dozens of servos (ie: see risk assessment).

The second milestone will improve our already existing robot, by adding sensors, and implementing more interesting gait patterns. Sensors will help us implement our robot's interaction with its environnment. This milestone will allow us to dive in more into the software side of our BeetleBot, notably coding the sensors, together with the servos.

The third milestone will also involve a software side, to make our bot responds to a remote input thanks to our wifi/bluetooth connective microcontroller. As this milestone is the last one,is quite challenging, and our bot should be functional by then, we consider this milestone as optional.

5.2 Work breakdown

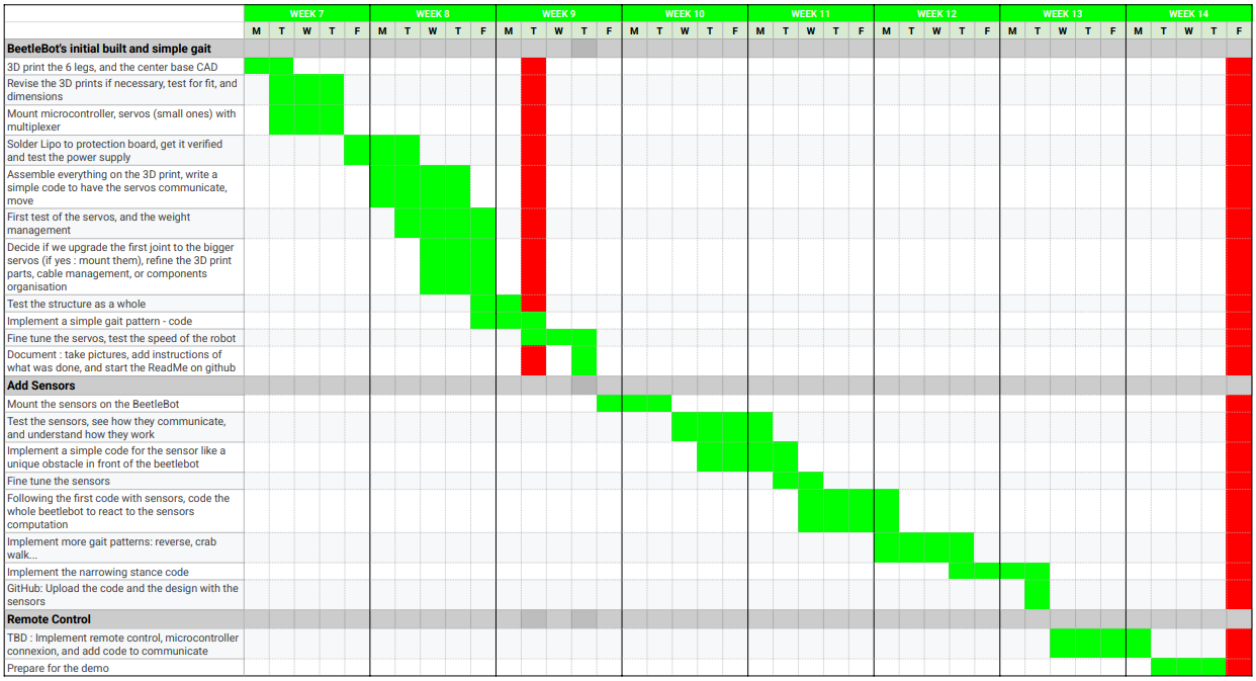

Figure 5: BeetleBot organisation chart

6 User stories

Biomechanics Researcher
As a biomechanics researcher, I want to analyze the hexapod's gait efficiency and compare it to biological arthropods to test hypotheses about energy optimization in locomotion. I want to study how different gaits affect energy expenditure, stability and speed. I can use the hexapod to validate hypothesis about insect movement. A similar study has been conducted by EPFL researchers.

CS358 Teacher
As an instructor, I want the hexapod to serve as a modular platform for teaching robotics principles, enabling students to experiment with gait algorithms, sensor integration, and collaborative problem-solving. I can challenge my students to improve the gait efficiency or implement terrain adaptation using sensor feedback.

Robotics Engineer
As a robotics engineer, I want to prototype and test new gait algorithms to improve the hexapod's adaptability to unstructured terrains. I can test the servo durability, torque and energy consumption under load to simulate real-world scenarios.

7 Future of the BeetleBot

The BeetleBot project can serve as a strong foundation for future research and development in the field of bio-robotics. By building upon its existing capabilities, we can expand its functionality, enhance its real-world applications, and push the boundaries of biologically inspired robotics. For example, the BeetleBot's current laser distance sensors provide basic obstacle detection, but integrating computer vision could revolutionize its environmental awareness. By adding lightweight cameras (e.g., Raspberry Pi) and leveraging machine learning, the robot could:

- Identify and classify objects (e.g., differentiating between obstacles, tools, or targets).
- Adapt its gait dynamically based on terrain (e.g., stepping over gaps or adjusting stride on slippery surfaces).
- Perform SLAM (Simultaneous Localization and Mapping) for autonomous navigation in unknown environments.

Another improvement can be made in regards to the motion of the hexapod. The current gait system relies on pre-programmed inverse kinematics, but future iterations could use reinforcement learning (RL) to:

- Self-optimize movement efficiency by testing different gaits and learning which ones minimize energy consumption.
- Adapt to damage (e.g., if a leg is impaired, the AI could recalibrate its walking pattern).
- Simulate bio-inspired behaviors (e.g., cockroach-like scrambling over obstacles or ant-like cooperative carrying).

Furthermore, the current BeetleBot design uses SG90 micro servos for most joints, prioritizing cost and weight savings. However, upgrading to higher-torque servos (e.g., MG996R or Dynamixel AX-12A) could unlock significant improvements in movement quality and durability. This would unlock many new capabilities and functionalities, such as stronger torque for smoother motion, longer lifespan, support for heavier payloads (such as ithe integration of additional sensors), but this would require a redesign of the current body, as it would need to accomodate the bigger servos and potentially a bigger battery.

8 References, inspiration

We have drawn some inspiration from existing Hexapods online, and tried to build our own project out of it. Some of those references include:

[1] Hexapod using SG90 motors
[2] Complex Hexapod using MG996R motors
[3] Code computation references, inverse kinematics
[4] Inverse kinematics, leg design