Federal University of Technology - Paraná – UTFPR
Academic Department of Electronics – DAELN
Academic Department of Informatics – DAINF
Computer Engineering
Integration Workshop 3 (ELEX22) – S71 – 2025/1

---

Technical Report

# SMAT: Stop Motion Animation Table

Daniel Augusto Pires de Castro– danielcastro@alunos.utfpr.edu.br
Guilherme Correa Koller– koller@alunos.utfpr.edu.br
Mateus Filipe de Ornelas Rampim– mateusrampim@alunos.utfpr.edu.br
Pedro Henrique Fracaro Kiche– pedrokiche@alunos.utfpr.edu.br
Thaís Say de Carvalho– thaiscarvalho@alunos.utfpr.edu.br
Victor Gabriel Almeida e Almeida– victoralmeida@alunos.utfpr.edu.br

June 2025

**Abstract**

The SMAT (Stop Motion Automation Table) project is an automated system designed to assist and optimize the stop-motion animation process. This paper presents the development of SMAT, covering its mechanical design, electronic implementation, and software architecture. The mechanical system features an XY table for precise puppet positioning and a servo-driven articulation mechanism. The electronic design includes power distribution, motor drivers for stepper and servo motors, and a Raspberry Pi as the central control unit, communicating with a remote PC-based application via a dedicated network connection. The software component consists of embedded control scripts running on the Raspberry Pi and a graphical user interface (GUI) developed for the PC, allowing animators to configure movements, trigger frame captures, and preview animations in real time. Additionally, the system integrates computer vision algorithms for hand detection and background removal, streamlining the post-production process. The final output includes an automated MP4 export of the captured animation sequence. The SMAT project offers a practical and efficient solution for animators, reducing manual workload and increasing the precision and speed of stop-motion productions.

## 1 Introduction

Stop Motion is an animation technique in which objects and characters are moved and their positions are captured frame by frame by a camera. By placing the frames in sequence, an illusion of movement is produced in the video rendering to animate these objects. The process is usually slow and time-consuming, even when using modern tools such as 3D printed object models or green screen backdrops for chroma key removal. Therefore, both production and post-production

require significant control and precision from the animators to obtain the desired results. This can harm the creative process of animators, as the time they could use to create their stories is consumed by developing these technical issues, especially with tight deadlines in the animation industry.

## 1.1 Proposed solution overview

The problem is solved by the SMAT: Stop Motion Animation Table, which is presented by the Figure 1. The project allows an animator to use an editing tool to determine movement instructions for their animation puppet. The movements are made automatically by the motors of the table and the puppet, and are then captured by a smart camera that detects whether the user intends to intervene or not. The animation is then processed according to all the user commands and can be edited to generate a final video.

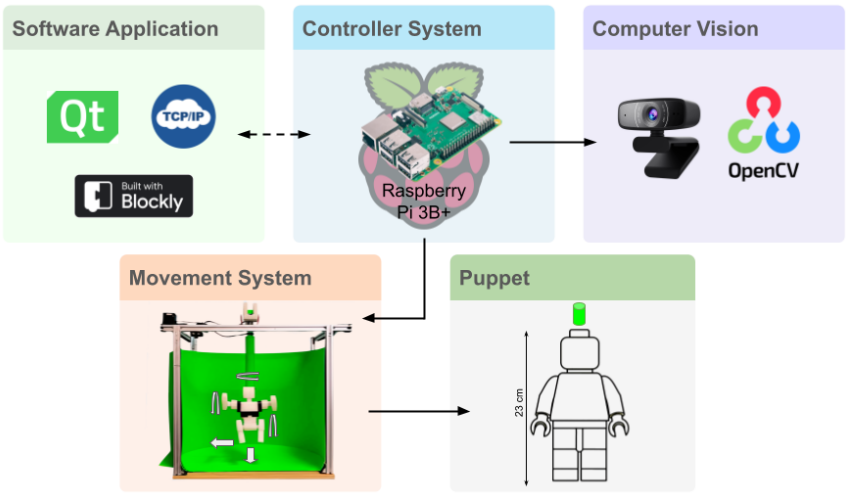


Figure 1: SMAT overall solution. Source: Authors (2025)

Initially, the user must use an application developed by the Python qtGUI library, generating sequences of movements through command blocks implemented in blockly. These instructions are then sent to the Raspberry Pi microcontroller and a TCP/IP communication is automatically established to send images produced by the robot's movement. Throughout the process, the camera detects if there is external movement, caused by hands or any object that appears on the edge of the image. If this occurs, the sequence of movements is paused and resumed only when there is no more intrusion detected by the camera. Finally, the animator exports the video and edits it according to his/her convenience.

The movement is done by means of an XY table of stepper motors connected to the microcontroller, together with the servomotors of the robot's joints. Their

positions are calibrated to respect the user's commands so that the camera captures movements that are consistent with the illusion of continuity. The XY table uses aluminum bars and contains a mechanical arm, all hidden by green screens to produce the removal of the background. The filaments pass through this mechanical arm and allow the robot to rotate without causing breakage. The robot is humanoid, is 23 cm tall and was 3D printed, allowing the animator freedom to customize it according to their design interests.

## 2 Project Specification

The project was based on a total of 92 requirements, corresponding to the important functionalities for implementing SMAT. They were necessary to describe the development stages, divided into Hardware, Mechanical and Software (of the application and the embedded system), among them functional and non-functional. In table Table 1 you can see the main requirements, for a general understanding of the system. In the project blog, available online at the Notion URL, you can check all the requirements, even those chosen as optional. In addition, it was necessary to integrate the requirements using a scheduler to link them to tasks throughout the weeks of deliverables.

| **Req.** | Description |
|---|---|
| **FR07** | The application must let the user define robot movements by instructions. |
| **FR10** | The application must send to the microcontroller the instruction of the moviment. |
| **FR11** | The application must get the images from the microcontroller. |
| **FR13** | The application must set the time of the images merged display according to the frame rate instructed by the user. |
| **FR14** | The application must show the resulting video of the instruction. |
| **FR05** | The application must allow using a custom image as background. |
| **FR25** | The microcontroller must pause movements when a hand is detected. |
| **FR28** | The microcontroller must capture a picture from the camera once the movement to the frame is completed. |
| **FR35** | The hardware must include a power supply capable of powering all motors and controllers safely. |
| **FR42** | The XY table must be connected to the puppet by its head. |
| **FR48** | The system must include a table board for support the puppet and the XY table. |
| **NFR84** | The hand detection must be made by detecting the edges of the picture, so when a hand or other objects (clamp, pincers) appears from the edges, the video will be stop. |
| **NFR54** | The puppet must be made of 3D filament. |
| **NFR58** | The system must have 2 Stepper motors (NEMA 17 or equivalent) for X and Y axis control. |
| **NFR59** | The system must have 6 SG90 micro servos for puppet articulation (arms, legs, neck, rotation). |

Table 1: Main project requirements. Source: Authors (2025). | **FR-**Functional Requirement | **NFR-**Non Functional Requirement Requirement

SMAT was limited to the fact that the animator could only use one movable puppet in the scene, allowing the robot to interact with other objects by simply placing and moving them during recording. For this reason, the features for detecting intrusions in the scene and removing the green screen were essential, as they allow customizations to the scene desired by the animator. The possibilities for specifying the frame rate follow conventional animation standards — between 10 and 24 FPS, and the entire process is automated to convert it into a frame of movement. However, the interface did not have all the conventional video editing tools, such as video viewing timelines and frame manipulation. Even so, the project has the essential basic features to make unlimited anima-

tion scenarios possible for the user to create.

# 3 Development

The project was developed in parts, divided into Mechanics, Electronics and Software. For each of these parts, there was a preliminary design in order to plan the necessary measures for implementing SMAT. For this reason, as one part was produced, the next part of the project was designed, until reaching the final integration of everything.

## 3.1 Mechanical Structure

### 3.1.1 Puppet

The puppet was initially designed to have a humanoid body with holes to support the servomotors in its arms and legs. On top of it, there is a coupling to connect it to the mechanical arm of the XY table, which contains another motor to rotate the robot around its own axis. There was a previous intention to allow movement of the robot's neck, but this requirement was made optional and was not implemented due to the difficulty with coupling its axis. All of this was designed to allow the most basic possible movements integrated with the electronic components.

To design the puppet, it was necessary to use Fusion 360 [1], a 3D modeling software that has useful features for mechanical projects. In the Figure 2 you can see the puppet design. The arms are tilted to give the impression of walking, since this is the basic movement it can make. Furthermore, it was not necessary to care for the robot to be balanced while standing, since it is naturally suspended, being held by the mechanical arm of the table. In the end, the puppet was 3D printed using PETG filament. In the Figure 3 it's possible to check the result of the 3D printing of the puppet.

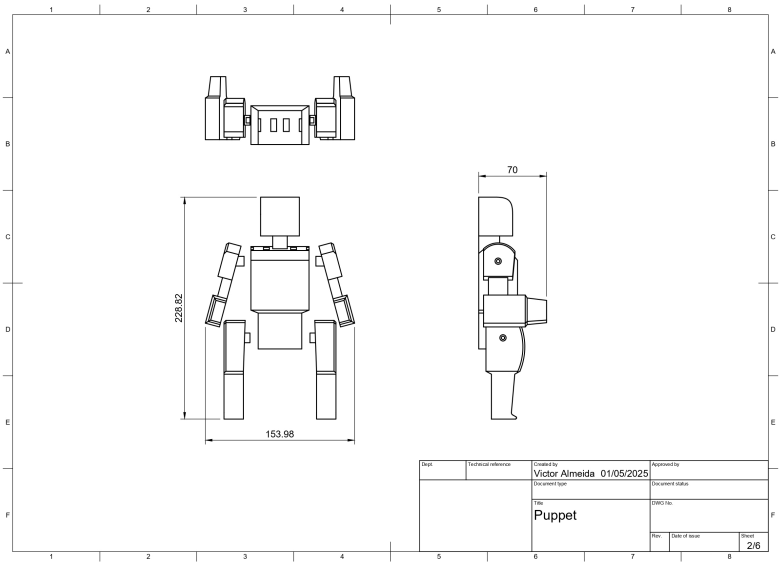


Figure 2: Puppet Drawing. Source: Authors (2025)

Figure 3: Printed Robot. Source: Authors (2025)

### 3.1.2 XY Table

Fusion 360 software was also used to model the XY table, as seen in the drawing of the Figure 4, which includes the aluminum bars and the arrangement of the entire rail system with skates and balls (known as linear guides) to allow movement with high precision and low backlash. This support integrates with the

robot through the coupling and has the built-in electronics to capture the commands from the microcontroller.

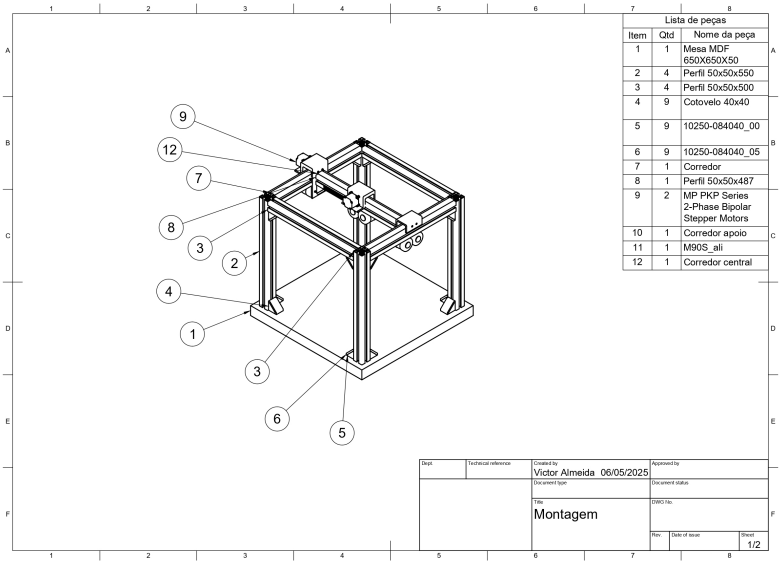


Figure 4: XY Table Drawing. Source: Authors (2025)

A MDF board with the dimentions 45cmX65cm supports all of the aluminium rails and includes a space for the support of the microcontroller and the circuit. It can be verified in Figure 5. The XY table has a coupling for conecting the axis rotation motor for the puppet, as described in Figure 6.

Figure 5: XY Table Assembly. Source: Authors (2025)

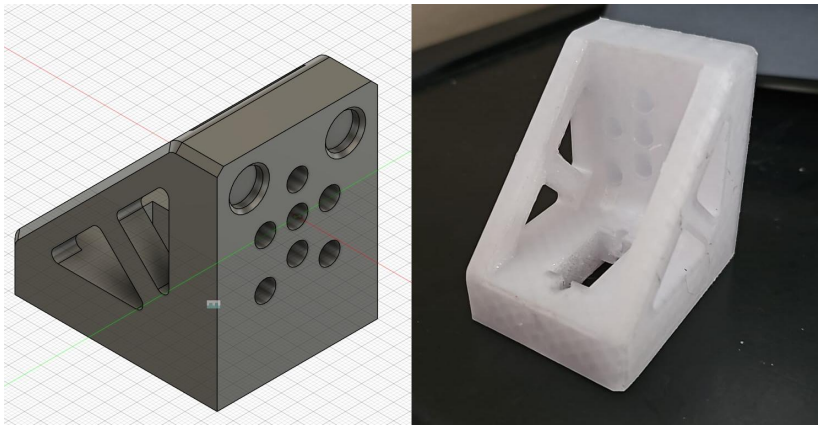

Figure 6: XY Table Coupling. Source: Authors (2025)

## 3.2 Electronical

All the circuit is inside a black patola box, as described in Figure 7.

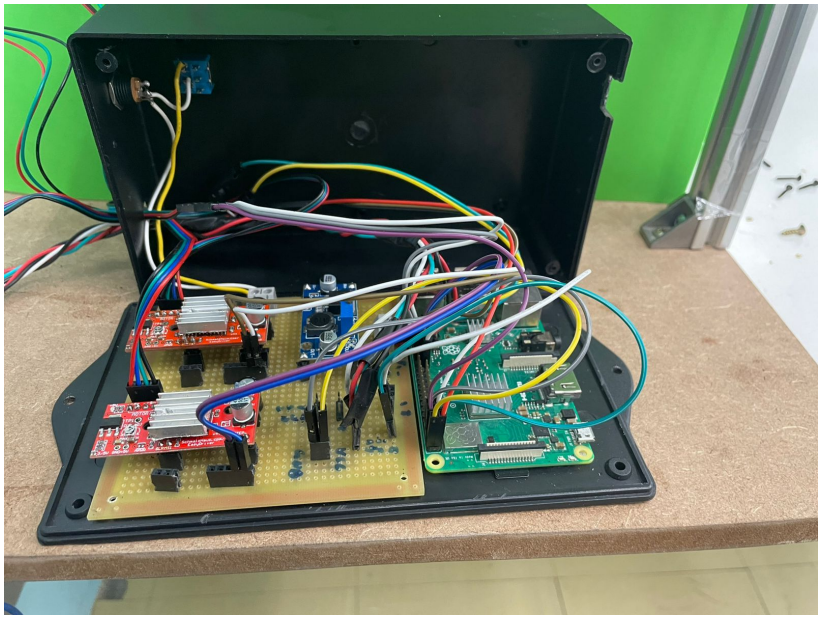

Figure 7: Patola Box for the Circuit. Source: Authors (2025)

### 3.2.1 Main Connection Board

The first hardware component designed was the main connection board, whose schematic is presented in Figure 8. This module acts as the central interface between the Raspberry Pi controller [2] and the system's peripherals. It groups the necessary GPIO signals for the stepper motor control, exposes the I²C communication bus for the servo motor module, and provides connection points for other critical sensors. Specifically, the end-stop sensors for the XY table are connected directly to GPIO 13 and GPIO 19, allowing the system to detect the physical limits of the axes. Additionally, the board manages the power distribution to the different components of the project, facilitating the assembly and ensuring a robust connection between the modules.

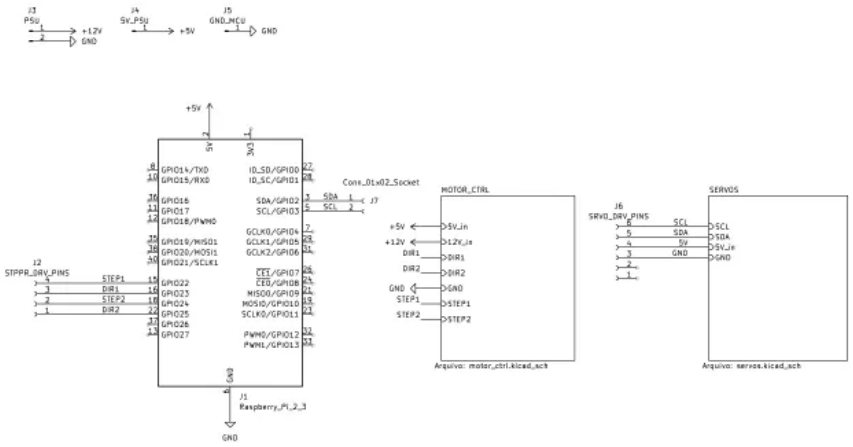


Figure 8: Electrical schematic of the main connection board. Source: Authors (2025)

### 3.2.2 Stepper Motor Control Module

The stepper motor control module, detailed in the schematic of Figure 9, is responsible for driving the two bipolar NEMA 17 motors that move the table on the X and Y axes. The circuit uses two A4988 drivers [3], which receive the step (STEP) and direction (DIR) signals from the Raspberry Pi and convert them into the necessary current to move the motors with high precision. To ensure stable operation and suppress voltage spikes from the motors, each driver's power input is supported by a 100µF electrolytic decoupling capacitor. This setup allows for the exact positioning of the platform in each frame of the animation.

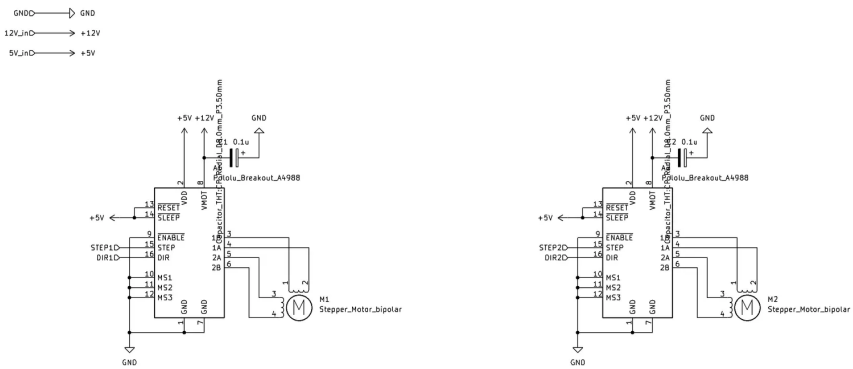


Figure 9: Electrical schematic of the stepper motor control module. Source: Authors (2025)

### 3.2.3 Servo Motor Control Module

For the control of the puppet's multiple articulations, a servo motor control module was designed, as presented in Figure 10. This circuit is based on the PCA9685 driver, a 16-channel PWM controller that communicates with the Raspberry Pi via the I²C bus. This approach is highly efficient, as it allows controlling up to 16 servo motors using only two GPIO pins from the microcontroller (SDA and SCL). The PCA9685 autonomously generates the stable and precise PWM signals required for the positioning of each joint, freeing up the Raspberry Pi for other tasks.

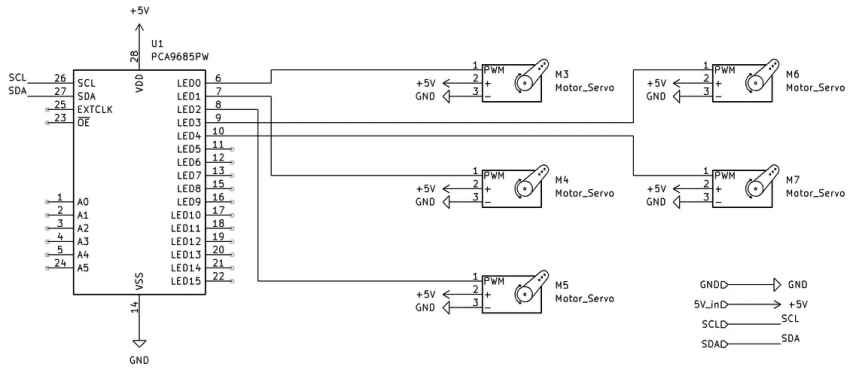


Figure 10: Electrical schematic of the servo motor control module. Source: Authors (2025)

### 3.2.4 Power Supply

The system's power supply strategy was designed to ensure stable and safe operation for all electronic components. As indicated on the main board (Figure 8), the system is powered by external voltage sources and uses two distinct power lines:

- **12V Supply:** Intended exclusively for the stepper motor drivers (A4988). This higher voltage is necessary to ensure adequate torque and performance from the stepper motors during the movement of the XY table.
- **5V Supply:** Responsible for powering the Raspberry Pi microcontroller, the PCA9685 servo controller, and the servo motors themselves.

This separation ensures that the microcontroller and the more sensitive control circuits receive a regulated and stable voltage, without being affected by the electrical noise and voltage drops that can be generated by the high-current switching of the stepper motors.

### 3.2.5 Peripherals and Utilities

The complete hardware design for SMAT integrates key off-the-shelf peripherals and utilities to fulfill its operational requirements in a modular fashion. These components are connected to the main Raspberry Pi controller to perform tasks ranging from precise motor control to image capture. The main peripherals are:

- **[1x] USB Webcam/Phone Cam:** Connected directly to the Raspberry Pi's CSI interface, this module acts as the system's "eye." It serves the dual purpose of capturing high-resolution frames for the animation and providing the video feed for the software's intrusion detection feature.
- **[2x] Stepper Motor Driver (A3967):** Each of the two NEMA 17 motors of the XY table is controlled by a dedicated driver [4]. These modules interpret the low-voltage logic signals from the Raspberry Pi and convert them into the high-current power required for the precise, step-by-step movement of the motors.
- **[1x] 16-Channel Servo Driver Board (e.g., PCA9685):** To control the six SG90 servo motors of the puppet's joints without overwhelming the Raspberry Pi's GPIO pins, an I²C servo driver board is used. This utility offloads the signal generation task, providing stable and precise control over all puppet articulations.
- **[1x] Power Supply Unit (PSU):** A dedicated power supply is a critical utility to ensure stable and safe operation. It is sized to provide sufficient and reliable power to the Raspberry Pi, the high-torque stepper motors, the servo motors, and all other electronic components simultaneously.

Considering all that was presented in this section, the final internal hardware assembly of the SMAT project is represented in the Figure 11.

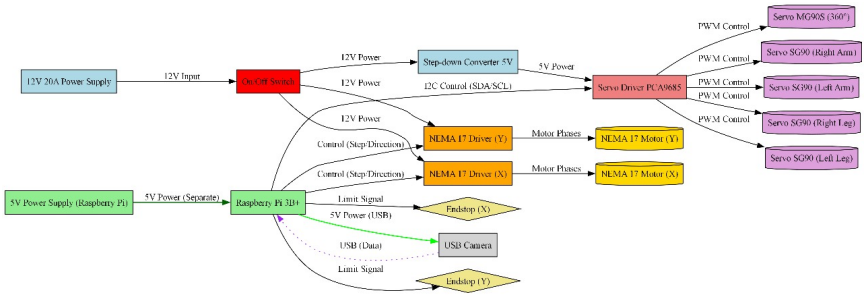


Figure 11: SMAT internal hardware assembly diagram. Source: Authors (2025)

## 3.3 Software

The software component of SMAT is the bridge between the animator's creative intent and the physical execution by the hardware. It is architecturally divided

into two distinct but interconnected parts: the embedded software running on the Raspberry Pi controller and the user interface application running on the animator's computer. These two components communicate over a TCP/IP network, creating a responsive and powerful system for creating animations. This section will present each of these parts in detail.

### 3.3.1 Embedded Software

For a better visualization of the software architecture, Figure 12 shows the designed class diagram for the embedded system.

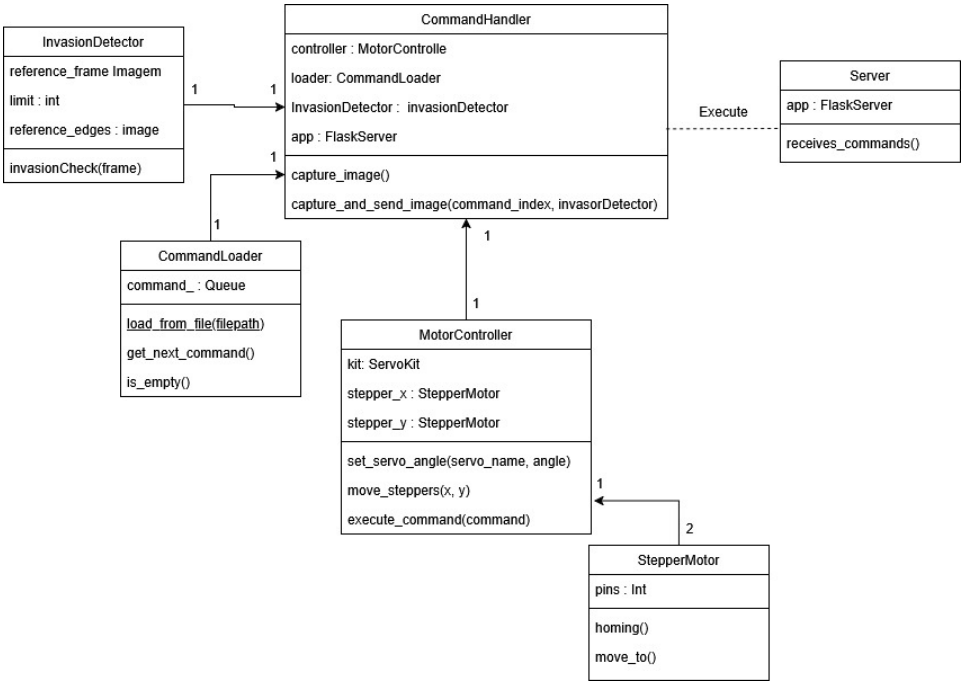


Figure 12: Embedded software class diagram. Source: Authors (2025)

As seen in Figure 12, the SMAT embedded software is structured around a central handler and a set of supporting classes responsible for motor control, command loading, invasion detection, and server interaction.

The CommandHandler is the main module of the system. When the server receives a new 'comandos.json' file, this component is triggered. It loads and executes each command sequentially, captures images from the webcam, verifies for intrusions, and sends the validated frames back to the desktop interface.

There are several supporting classes that provide the core functionality:

- **MotorController:** This class handles all movement-related tasks. It instantiates and controls both stepper motors (X and Y axes) and servos (for

limb movements). It provides methods to execute full command instructions and low-level motor operations.

- **StepperMotor:** Represents a stepper motor, storing configuration parameters such as pin assignments and providing methods like `homing()` and `move_to()` to move to specific positions.
- **CommandLoader:** Loads animation instructions from a JSON file and stores them in a queue. Commands can be fetched one by one using `get_next_command()`, and `is_empty()` is used to determine if all commands have been processed.
- **InvasionDetector:** Compares each new image captured from the webcam with a reference image to check for unwanted movement at the frame borders. It was inspired by other project which uses AI for hand detection [5]. However, SMAT uses edge detection and dilation to detect any significant changes and prevent capturing intruded frames.
- **Server:** A Flask-based web server responsible for receiving the 'comandos.json' file from the desktop application. Upon receiving a file, it saves it locally and triggers the execution process through the `CommandHandler`.

This architecture ensures modularity and maintainability, allowing each part of the system to be tested independently while coordinating seamlessly during the animation playback process.

### 3.3.2 User Interface Application

The SMAT user interface is a desktop application developed with Python and a Qt-based GUI library. It provides all the necessary tools for an animator to create, preview, and export a stop motion animation. Figure 13 illustrates the main classes of the application.

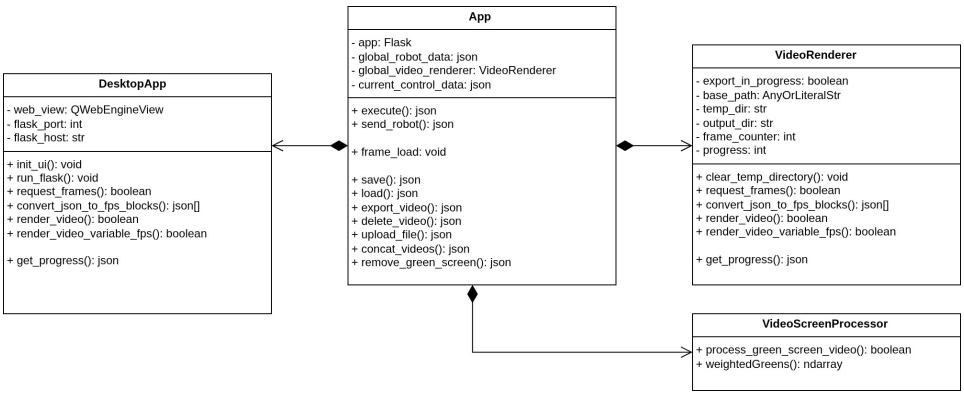


Figure 13: SMAT Application Class Diagram. Source: Authors (2025)

The application communicates with the SMAT table via a TCP/IP connection, sending animation instructions and receiving captured frames. This is done by creating Flask servers and clients within the application and the Raspberry Pi. The python library requests is responsable for making the request. The primary data sent is the entire animation sequence, structured as a JSON array. Each object in the array represents a single frame and contains the absolute target position for each of the 7 motors.

The animation sequence is a list of frame objects. The example below shows two frames from a sequence where only the Y-axis is moving.

```
[
  {
    "BDireito": 0,    // Servo position for Right Arm
    "BEsquerdo": 0,   // Servo position for Left Arm
    "OwnAxis": 0,     // Servo position for Puppet Own Axis
    "PDireito": 0,    // Servo position for Right Leg
    "PEsquerdo": 0,   // Servo position for Left Leg
    "X": 0,           // Stepper position for X axis
    "Y": 0            // Stepper position for Y axis
  },
  {
    "BDireito": 0,
    "BEsquerdo": 0,
    "OwnAxis": 0,
    "PDireito": 0,
    "PEsquerdo": 0,
    "X": 0,
    "Y": 5.555555
  }
]
```

The application receives status messages and image data, so the user is able to receive feedback on how the process is going. When an error occurs, a popup appears to indicate what happened. The updates are done by simple JSON messages indicating progress. When a picture of each frame is catched, the Pi sends it to the application. A folder containing temporary frames is filled with the new ones and the User is able to export it, so the "frame_captured" status is shown.

The application is built around a single main window that consolidates all tools needed for the animation workflow. The interface, shown in its entirety in Figure 14, is divided into three main functional areas.

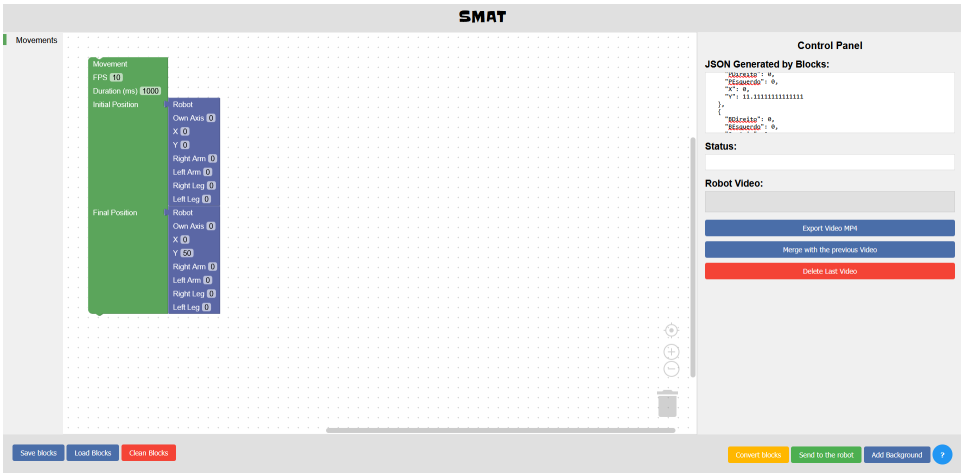


Figure 14: The main graphical user interface of the SMAT application, showing the Blockly workspace (left), the Control Panel with JSON preview and video options (right), and the main action bar (bottom). Source: Autors (2025)

The largest area of the interface is the visual programming workspace, labeled "Movements" and implemented with Blockly [6]. The animator defines motion segments using "Movement" blocks, as exemplified in Figure 15. Within these blocks, the user sets start and end keyframes by entering the numerical target positions for the XY table and each puppet servo into the nested "Robot" position blocks. The user also sets the FPS and duration parameters that the software will use for the interpolation calculation. Below this workspace, buttons for "Save Blocks", "Load Blocks", and "Clean Blocks" provide project management capabilities.

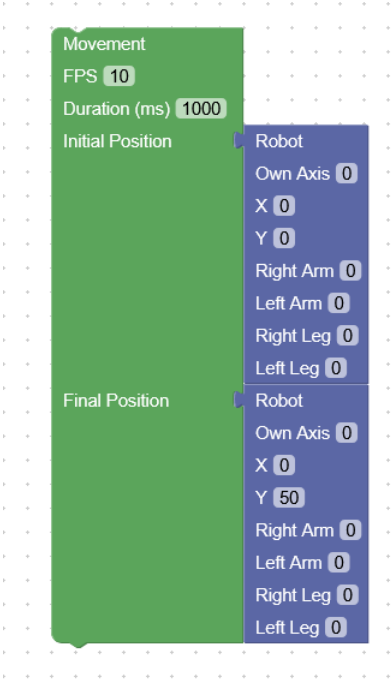


Figure 15: Example of the main 'Movement' instruction block. The user defines the start and end poses (keyframes) by entering numerical values directly into the nested 'Robot' position blocks. Source: Autors (2025)

In the right of the interface, there is the Control Panel, that provides real-time information, process monitoring, and output controls. It is subdivided into:

- **JSON Generated by Blocks:** A text area that displays the machine-readable JSON array generated from the visual blocks. This allows the user to verify the interpolated frames before sending them to the robot. This area can be edited by the User.
- **Status:** A display field that shows feedback and status messages received from the SMAT table during operation (e.g., "frame_captured", "paused").
- **Robot Video:** A set of functions for video post-production. This includes the "Export Video MP4" button to create the final animation file, a "Merge with the previous Video" button to stitch multiple animation segments together, and "Delete Last Video". For video editing, the main python library used was OpenCV [7].

In the Right Bottom of the interface, there is the Action Bar, which is a row of primary action buttons that drive the main workflow.

- **Convert Blocks:** Generates the sequence of interpolated frames from the keyframes defined in the workspace and displays them in the JSON preview area.

- **Send to the robot:** Transmits the generated JSON sequence to the SMAT table to begin the automated capture process.
- **Add Background:** Allows the user to select a custom background image for the scene. It was based on a chroma key algorithm [8] applied as seen in Figure 16.
- **Helper Manual**: Includes a '?' button, to ensure for the User the main instructions of uusing the Application.

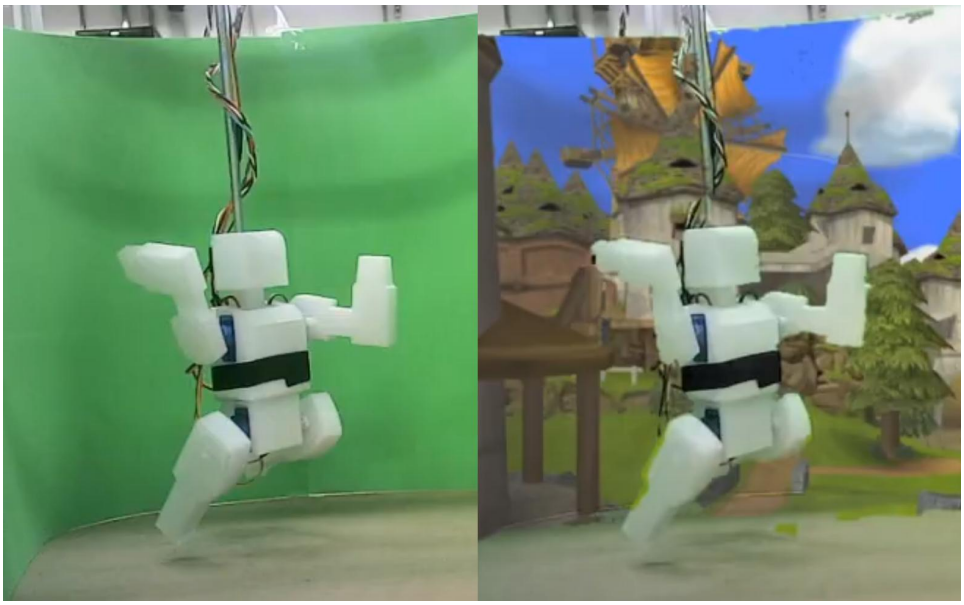

Figure 16: Background Removal. Source: Authors (2025).

# 4 Results

## 4.1 Budget

The total estimated budget for the SMAT project was structured to include both the core components and a contingency fund for risk mitigation. The primary components required for the initial assembly amounted to R$ 1035.00. An additional R$330.00 was allocated for spare parts and unexpected events, such as replacing components or needing a second 3D printing run. This brought the total planned budget to R$ 1365.00, as detailed in Table 2.

Table 2: Project Budget

| **Item** | **Project Relation** | **Cost (R$)** |
|---|---|---|
| Raspberry Pi 3B+ | Original Estimation | 250.00 |
| USB Camera | Original Estimation | 60.00 |
| Micro servo MG90S (360 degrees) | Original Estimation | 35.00 |
| Micro Servo SG90 (5 units) | Original Estimation | 50.00 |
| 3D printed Robot | Original Estimation | 50.00 |
| Servo Driver PCA9685 | Original Estimation | 25.00 |
| NEMA 17 Stepper Motor (2 units) | Original Estimation | 120.00 |
| Stepper Driver A3967(2 units) | Original Estimation | 20.00 |
| 12V 20A Source | Original Estimation | 30.00 |
| Step-down converter | Original Estimation | 10.00 |
| Aluminum Rails | Original Estimation | 200.00 |
| MDF Wood | Original Estimation | 30.00 |
| Belts and Pulleys (2 units) | Original Estimation | 100.00 |
| Endstop Switches (2 units) | Original Estimation | 20.00 |
| Cables and Connectors | Original Estimation | 30.00 |
| On/Off Switch | Original Estimation | 5.00 |
| Spare SG90 servos (3 units) | Risk Mitigation | 30.00 |
| Spare budget for second printing | Risk Mitigation | 50.00 |
| 12V 20A Source (spare) | Risk Mitigation | 30.00 |
| NEMA 17 Stepper Motor (spare, 2 units) | Risk Mitigation | 120.00 |
| Extra budget for unexpected events | Risk Mitigation | 100.00 |
| **Total** | | **1365.00** |

## 4.2 Schedule

The project's timeline was organized into six distinct stages: initial development of the Mechanical, Hardware, and Software sections, followed by integration phases. The allocation of hours and the time spent on each stage are detailed in Table 3.

The data shows that the project was executed efficiently through the initial development and integration stages, with most phases being completed under the estimated time. However, during the final period leading up to the presentation, unforeseen and complex integration problems emerged. Solving these last-minute issues to ensure a stable and fully functional final product required a significant troubleshooting effort. This is reflected in the 350 hours spent in the final stage, far exceeding its estimate and causing the total project time to slightly surpass the initial plan.

| Name | Estimated time (h) | Spent time (h) |
|---|---|---|
| Mechanical Section | 145 | 125 |
| Hardware Section | 128 | 88 |
| Software Section | 122 | 95.5 |
| Integration | 135 | 134.5 |
| Final Integration | 164 | 136 |
| Presentation | 234 | 350 |
| **Total** | **928** | **929** |

Table 3: Project Schedule by Stage

Figure 17 illustrates the functional requirements achieved, categorized into mandatory done, mandatory partially done, optional done and optional not done. The uncompleted optional requirements includes only aesthetic features to enhance the design, such as moving the doll's neck and signaling LEDs. The partially completed feature was the puppet's rotation around its own axis, which does not complete the 360º turn.

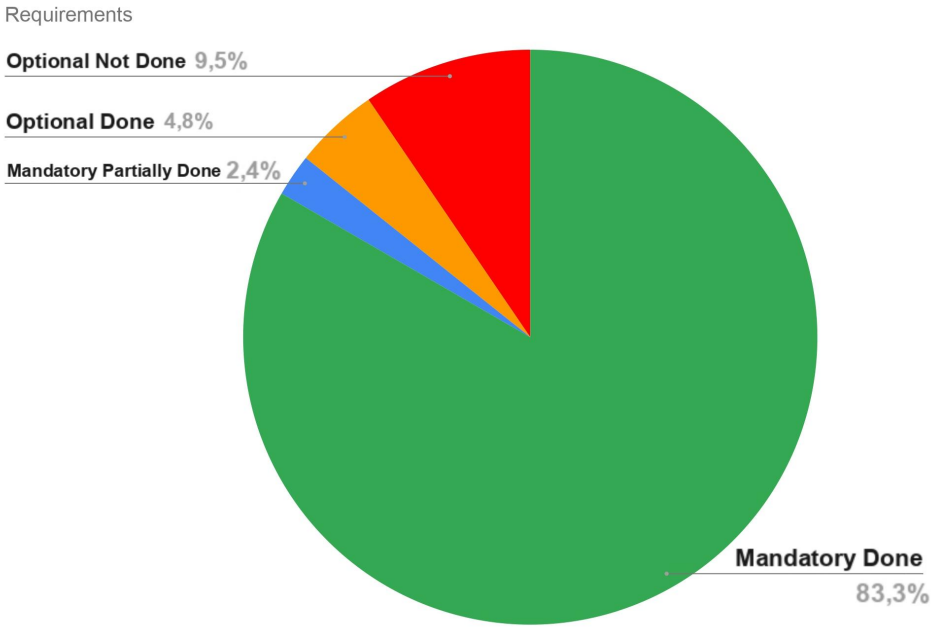


Figure 17: Functional Requirements Achieved. Source: Authors (2025)

## 5 Conclusions

The SMAT project has been a challenging and rewarding endeavor, representing nearly 1000 hours of design, development, and testing across mechanical, electronics, and software disciplines. It has successfully demonstrated its feasibility

as an effective proof of concept for an automated stop motion animation table. The results highlight its ability to reduce manual effort, streamline the animation process, and expand the creative possibilities available to animators.

Although certain optional features were not fully implemented, the project nonetheless met its core objectives and established a solid foundation for future improvements. The attention to detail in both hardware and software design ensures that SMAT can be further refined and adapted for a variety of applications. In the end, this work showcases teh potential for automated tools to transform traditional stop motion animation and stands as an acomplished example of engineering and innovation.

## References

[1] Autodesk. Fusion 360. https://www.autodesk.com/br/products/fusion-360/overview.

[2] Raspberry pi. https://www.raspberrypi.com/.

[3] Allegro MicroSystems. A4988 datasheet. https://www.alldatasheet.com/datasheet-pdf/pdf/338780/ALLEGRO/A4988.html.

[4] Allegro MicroSystems. A3967 datasheet. https://www.alldatasheet.com/datasheet-pdf/pdf/83571/ALLEGRO/A3967.html.

[5] Tom Nardi. Mastering stop motion through machine learning. https://hackaday.com/2021/09/20/mastering-stop-motion-through-machine-learning/, 2021.

[6] Google Developers. Blockly. https://developers.google.com/blockly?hl=pt-br.

[7] Open source computer vision library. https://opencv.org/.

[8] Bogdan Tomoyuki Nassu. Background chroma key removal. https://www.youtube.com/watch?v=rkQQw0wGay8, 2016.