The Machinations and Ethics of Autonomous and Non-autonomous Melty-Brain Combat Robots

by

John Aduroja

A thesis presented to the Honors College of Middle Tennessee State University in partial fulfillment of the requirements for graduation from the University Honors College

Spring 2026

Thesis Committee:

Dr. Vishwas Bedekar, Thesis Director

Dr. Ken Currie, Second Reader

Dr. John Vile, Thesis Committee Chair

The Machinations and Ethics of Autonomous and Non-Autonomous Melty-Brain Combat Robots

by John Aduroja

APPROVED:

Dr. Vishwas Bedekar, Thesis Director

Professor, Engineering Technology (ET)

Dr. John Vile, Thesis Committee Chair

Dean, University Honors College

Acknowledgement

This work was supported and funded by the Middle Tennessee State University's (MTSU) Honors College. Much gratitude to God for enabling the successful completion of this project and to Dr. Vishwas Bedekar for his valuable guidance, insights, and supervision. Special thanks to Mr. Robert W. Smith for his contribution and guidance in establishing the autonomous section of this project, Dr. Ken Currie for reviewing the initial and final thesis draft, and for serving as the second reader for this thesis, the Honors College faculty and advisors, Mr. Bereket Tegistesillassie, Mr. David Emeruwa, Mr. Jacob Farmer, Ms. Fiyinfoluwa Ayo-Oyalowo, Mr. Alaba Akintola, and Mr. Aidan Martin for their respective feedback and contributions towards this thesis. Finally, much thanks to my parents and siblings for their support throughout the course of this thesis.

Abstract

Melty Brain combat robots, also known as tornado or spinning-while-moving robots, are rare full-body spinners in which the entire machine revolves around its center of rotation (Beagle Bone). The robot's operation relies on a complex drive system that can be best described as translational drift. Functionality requires programming microcontrollers (the robot's brain) alongside sensors such as accelerometers, enabling mobility and spinning without a dedicated weapon.

This inquiry explores the Melty Brain robot as a unique case study in combat robotics, offering insight into both autonomous and non-autonomous driven systems. It examines technical machinations, risks, ethical considerations, and rational protocols to establish a framework for safe and precise integration of artificial intelligence. By situating AI as a key component of autonomous systems, the study highlights pathways for advancing combat robotics while informing broader applications in robotics research. Findings aim to support innovation, safety, and responsible development in this evolving field.

Table of Contents

List of Terms

Degree of Freedom: It represents the number of independent ways a system can move.

Brushless motors: They are high-efficiency electric motor that operates without mechanical brushes or a commutator, replacing them with electronic controls.

Kinematic Chains: They are an assembly of rigid bodies (links) connected by joints (pairs) that constrain relative motion, allowing them to transfer motion or force in a specific, predictable way.

Streamlining: designing or providing a form that presents very little resistance to airflow or water flow, increasing speed and ease of movement.

SolidWorks: It is a leading 3D computer-aided design (CAD) and computer-aided engineering (CAE) software application used by engineers and designers to create, simulate, publish, and manage 3D models and technical drawings

Bot: A term for a robot.

Integrated Circuits (IC): an electronic circuit formed on a small piece of semiconducting material, performing the same function as a larger circuit made from discrete components

Internet of Things (IOT): the interconnection via the internet of computing devices embedded in everyday objects, enabling them to send and receive data.

Revolution per minute (RPM): A unit of rotational speed, indicating how many full 360-degree rotations a machine component or object completes in one minute.

Virtual Heading: is a computed, synthetic, or reference direction that the robot's navigation system uses to determine its desired movement path, rather than relying solely on its physical orientation (the direction it is physically facing) or a compass sensor.

Kalman Filter: is an optimal estimation algorithm that predicts the state of a dynamic system (e.g., position, velocity) over time by combining noisy sensor measurements with prior knowledge.

Edge computing: It is a decentralized IT architecture that processes data near its source, such as IoT devices or local servers, rather than in a distant, centralized cloud or data centre.

Trillion Operations Per Second: measures a processor's speed by counting how many trillion basic mathematical calculations—such as addition or multiplication—it can perform in one second.

Thwack Robots:

Objectives

The initial objective is to innovate and build a three-pound (Beetle weight) combat robot. In this case, it will be a Melty brain combat robot that uses its entire mass as both a weapon and a means of motion, generating massive power and force to ensure effective and maximum destruction. Basically, adopting a typical and more common practice in combat robotics.

The subsequent and final objective, which is the core of this research's thesis, is to introduce automation into combat robotics. Essentially, ensuring our combat robot can operate with limited or negligible human interaction. This will entail a more programming-oriented approach to developing and building a combat robot, while highlighting its effectiveness, advantages, and potential limitations in a more competitive setting.

Introduction

In an ever-evolving world, concepts and fantasies from the previous decade, such as self-driving cars, drone delivery, and the Da Vinci surgical system (Intuitive), are becoming societal norms and realities, driven by the fundamentals and principles guiding the integration of robotics, particularly combat robots and AI.

These principles adhere to precise, accurate engineering in an evolving, controlled environment under high pressure and intensity. They guide biomedical robotics used in global standards of medical surgery and practice. The Melty brain combat robot combines these principles through precise control of rotational motion and body mass, which account for its material strength, mechanics, and durability, as both a weapon and a means of locomotion, through the combination of its microcontroller and accelerometer. Its uniqueness allows it to be used as both a tele-operated and a semi-autonomous battle Bot, requiring a transmitter to operate the robot manually. Finally, it has an autonomous system that enables the robot to operate autonomously.

Combat robotics is a sport-like classification of robotics that involves custom-built robots applied in diverse ways in an enclosed arena to disable or destroy their opponents. There are essentially multiple weight classes in the world of combat robotics, ranging from Flea weight (75 grams), Ant weight (1 pound), Beetle weight (3 pounds), and heavyweight (>60 pounds), with each class characterized by its size and power (Robo Table, 2025). These robots utilize a wide spectrum of weaponry, from high-speed spinners (vertical, horizontal, drum, and full-body types), that deal damage by impact and kinetic force, flippers and lifters that use a platform to throw opponents into the air or around, crushers and hammers designed for piercing armour and direct damage, to grapplers modified with a clamping mechanism to control

opponents. The melty brain combat robot is a high-speed spinner, otherwise known as thwack robots. They will be classified under the beetle weight category for this research, using the college-standard level of the National Robotics Challenge (NRC).

Robotic Design

Design is a crucial process for any combat robot across weight classes, as accurate optimization of durability, power, and strategy will determine the effectiveness and efficiency of a combat robot, and in this case, the Melty-Brain combat robot. The success of any battle Bot hinges on the smart, practical use of design, taking into consideration the material, weapon, and categories of the bot to give it an edge in the arena. This phase accounts for both robotic control and intelligence, aligning sensor, electronics, and weapon placement into the design to protect and properly utilize their accessibility for quick repairs.

Design Drafting

Design requires a comprehensive understanding of the problem at hand, the contexts in which it arises, and a resulting solution that addresses its specific nuances and intricacies (Galileo). This robot's design had to allow for locomotion while also using the motion generated to propel, with the intent of being used as a weapon. Therefore, a design that accommodates both the uniqueness of a translational drift drive and the full-body spinner is the solution for a Melty-Brain Combat robot.

The design drafted to address the need highlighted earlier for this bot was a circular compartment, as seen in Figure 1.1, streamlined to enable both linear and 360-degree rotational movement, used for drive and weaponry.

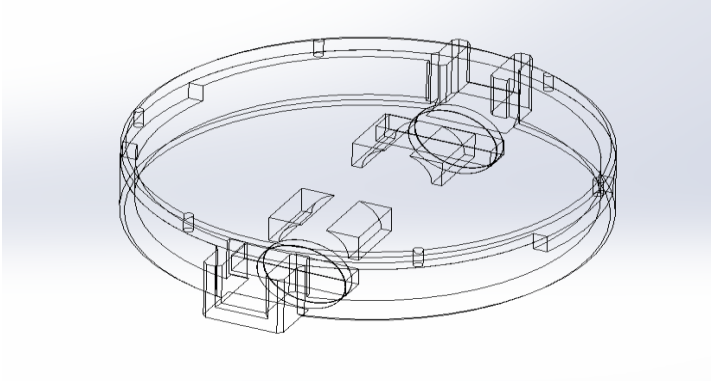


Figure 1.1 Melty-Brain Combat Robot's Initial Design (2025)

The dimension adopted for the circular body compartment was 10 inches in diameter, which is within the standard guidelines for the National Robotics Challenge of about 14 by 14 inches (NRC). A slot of about 0.9 inches was created to firmly attach the motors to the robot's wheels. Changes were made from the initial design in Figure 1.2a to the final design in Figure 1.2b, moving from a downward semi-circular design to an upward semi-circular design with a top of similar shape to serve as the motors' fasteners.

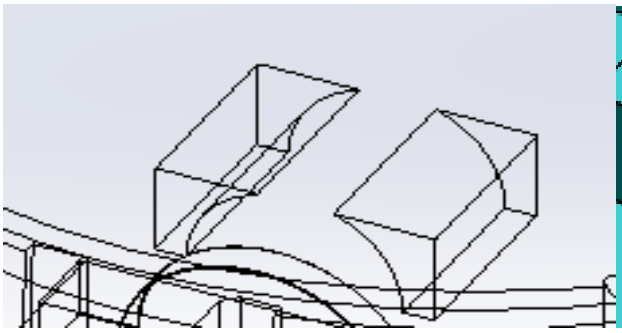


Figure 1.2a - Initial motor slot's draft.

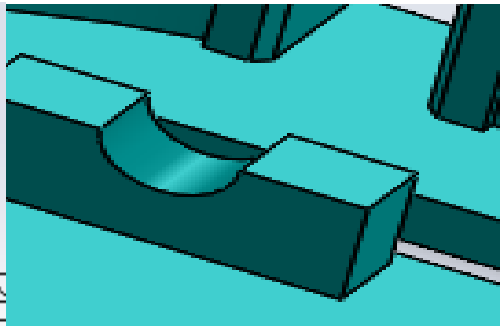


Figure 1.2b - Final motor slot's draft.

The robot weaponry system operates on a full-body spinner with body mass as the force and means of damage. In most combat robots, the weapon operates on a separate system from that of the drive system (locomotion). However, the Melty-brain robot simultaneously combines both drive and weapon systems, requiring a design that properly aligns the weapon with the body's center of gravity. Therefore, an oblong, hole-like structure about 3/8 inch long was created to fit the weapon. Initially, this design in Figure 1.3a was only introduced for the upper part of the body side compartment, resulting in a two-blade weapon. However, the robot's lack of protection at the lower part led to the addition of a slot, resulting in a four-blade weapon system in Figure 1.3b.

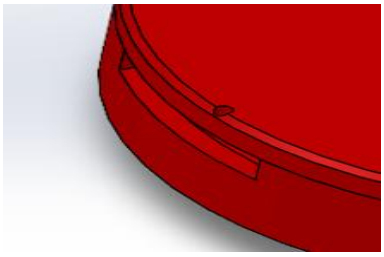


Figure 1.3a
Initial weapon
slot design

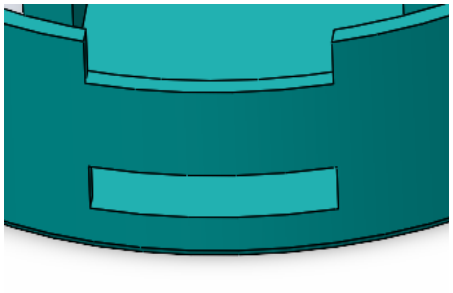


Figure 1.3b
Final weapon
slot design

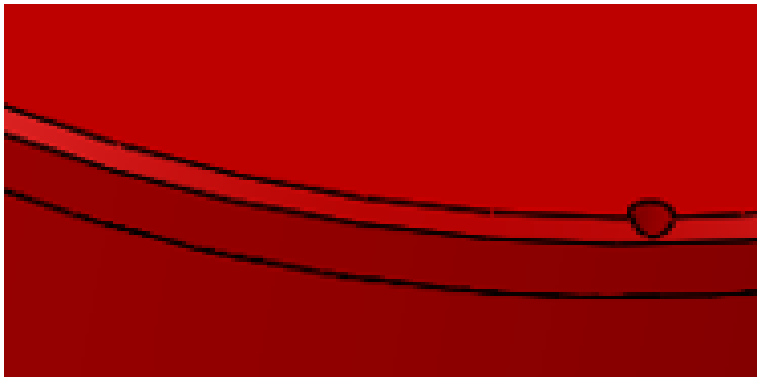
Material and Mechanical Structure

A robot's structure, alongside its kinematic chains, is crucial to its sturdiness and efficiency in the arena. Therefore, attention must be paid to the nuances and minute details of the robot's mechanical structure and materials, with respect to durability, strength, tensile capacity, ductility, and hardness.

Thermoplastic Polyurethane (TPU) filament was chosen for this Melty-Brain combat robot due to its exceptional flexibility, high elastic capacity, and vastly superior durability. TPU filament has an elasticity of about 200 to 1,000 percent before breaking; this, alongside its rubber-like properties, makes it ideal for absorbing the highly intense shocks and vibrations that the robot will be exposed to in a battle arena (MASSIVIT). This will protect and minimize damage to the robot's control elements and the "brain" (microcontroller), which are highly susceptible to shocks and vibrations. The TPU filament supports the robot's circular compartment design by providing flexibility, allowing for greater leverage for a full-body spinner without damaging the structure.

The circular compartment design of the robot's body is streamlined along its edges to enable smooth motion in all directions, especially given its unique shape. This is important because the robot moves in multiple directions in a battle arena, with its movement controlling both its drive and weapon system. Applying this aerodynamics principle (streamlining) is known to increase speed and enhance stability, which are essential for any combat robot in the arena (Mohan, Jatin, Vashisht). Figure 1.4 shows how this is used in the Melty Brain Combat Robot.

Figure 1.5
Aerodynamics Principle
(Streamlining)



The robot's final prototype was printed on a 3D printer, specifically the Bamboo X1E. This machine preheated the TPU filament to about 250 degrees Fahrenheit and molded it into the SolidWorks-designed shape. Figure 1.5a and 1.5b display the 3D printer used.

Figure 1.5a External view of Bamboo X1E Figure 1.5b Internal view of Bamboo X1E

Design Prototype

While designing the robot, both autonomous and non-autonomous approaches were considered based on their suitability for their respective purposes. Several factors were considered, including sensor slots, mass, locomotion, and weapon placement.

The autonomous design requires sensor slots, as they serve as the basis and means of interaction with the arena, from which the robot's decisions are inferred, highlighting the need for their placement. This slot is shown in Figure 1.6, with the detailed intricacies of its design. Furthermore, the necessity of a three-pound mass must be accounted for, as it is a beetle weight class (NRC). However, since it is an autonomous design, leeway is given, and it does not need to be exactly that weight. Weapon and drive placement were also accounted for, as discussed earlier in Figures 1.2 and 1.3.

The positioning of the sensors relative to the wheels was a key design factor, as it dictates the robot's logic and response times in software. The sensors were placed perpendicular to the wheels and the weapon blade to facilitate smoother rotation and allow the blade to build sufficient momentum before contacting the opposing robot.

The non-autonomous design prioritizes effective mass distribution over sensor integration, as the robot's radio-controlled nature eliminates the need for dedicated sensor mounting. This allows for more precise mass allocation compared to the autonomous version. Additionally, the non-autonomous build utilizes the same drive and weapon placement protocols as its autonomous counterpart to ensure mechanical consistency.

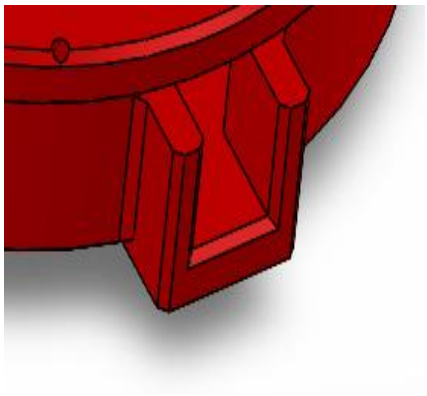


Figure 1.6 - Weapon slot

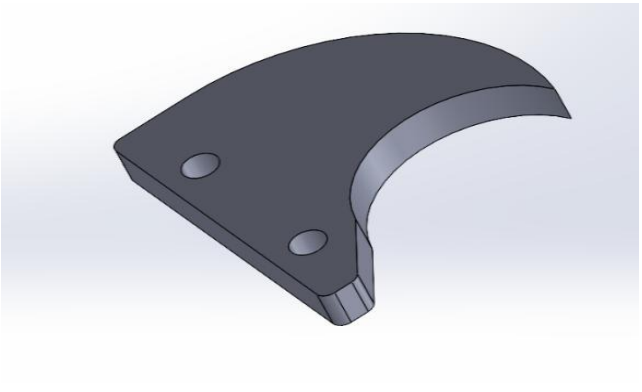


Figure 1.7 – Weapon blade (1/2 inches of aluminum)

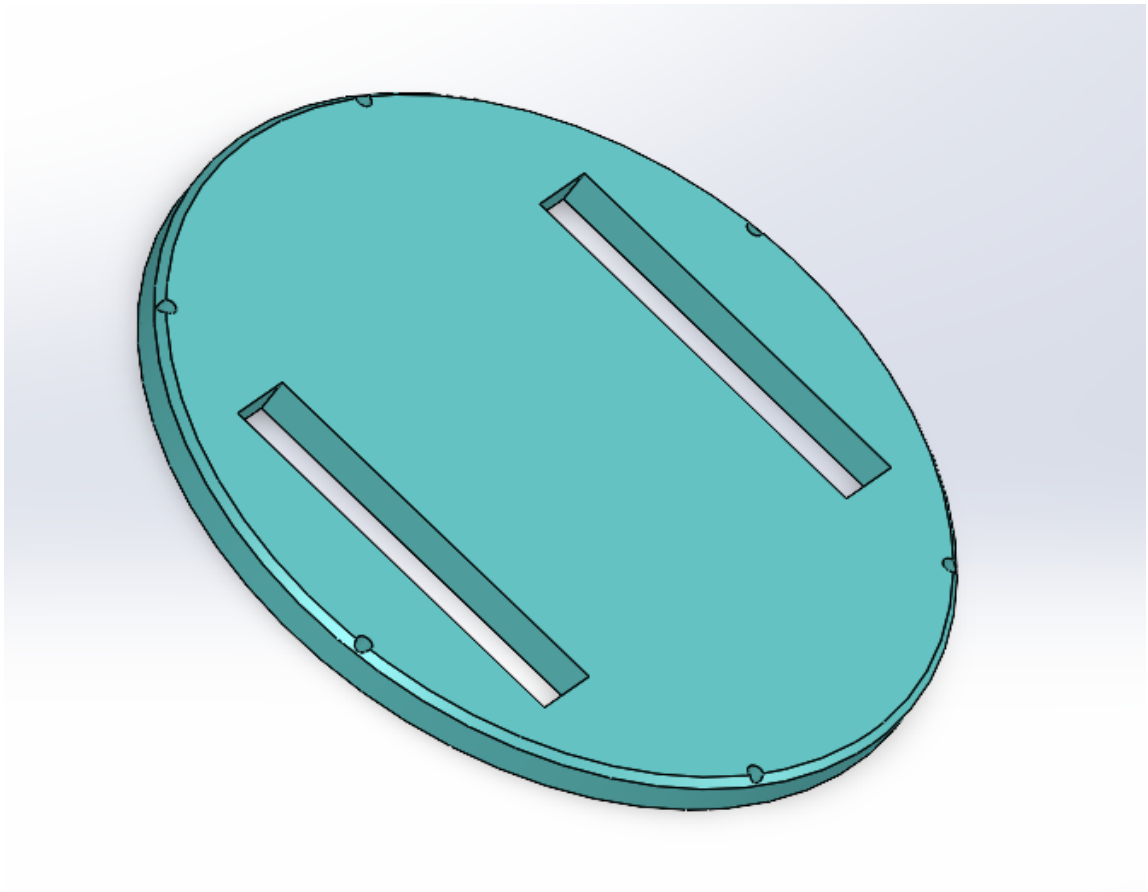
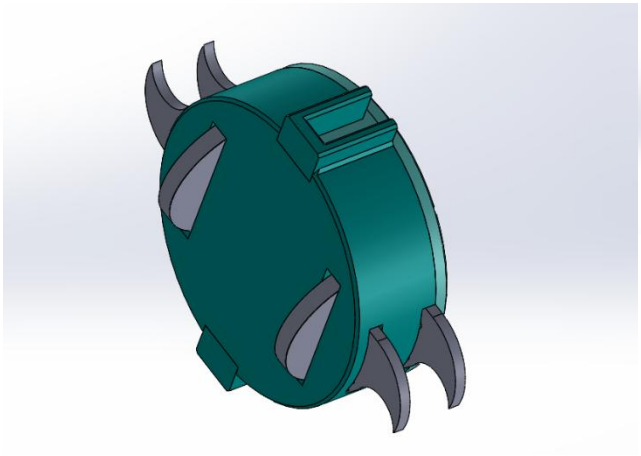
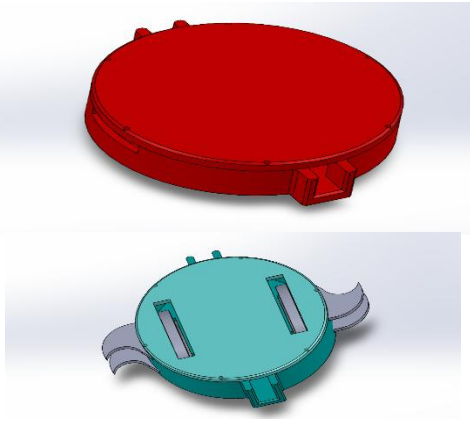


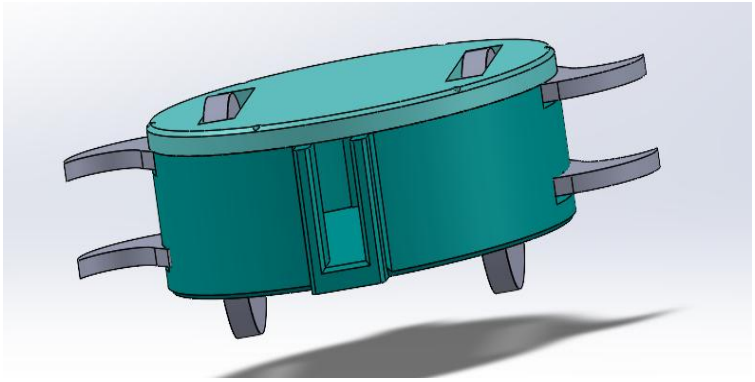
Figure 1.8 – Robot top (SolidWorks design)



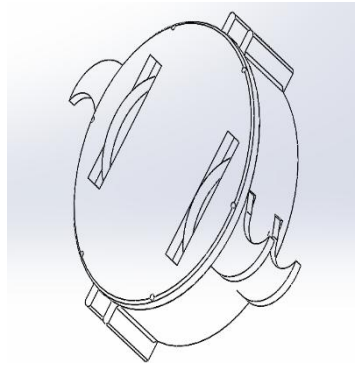
Bottom plate



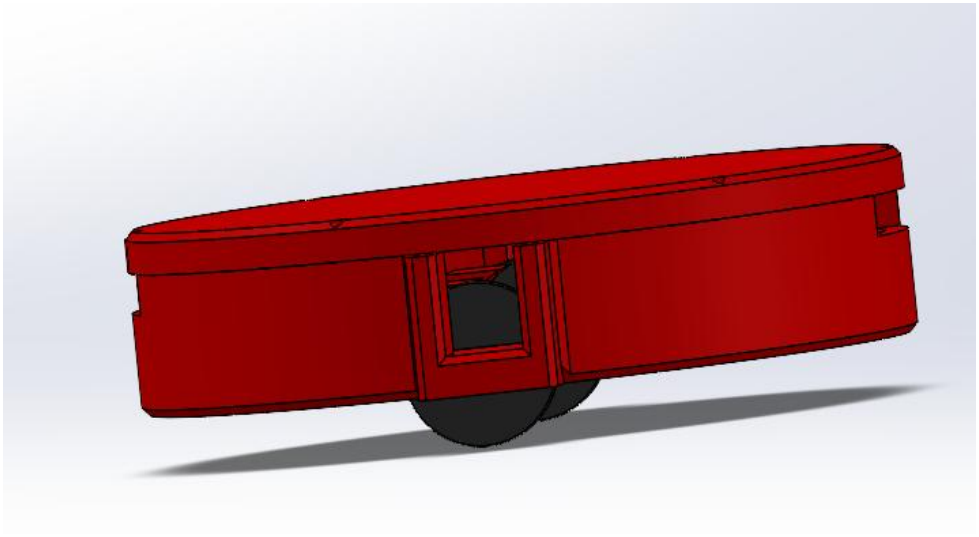
Top view



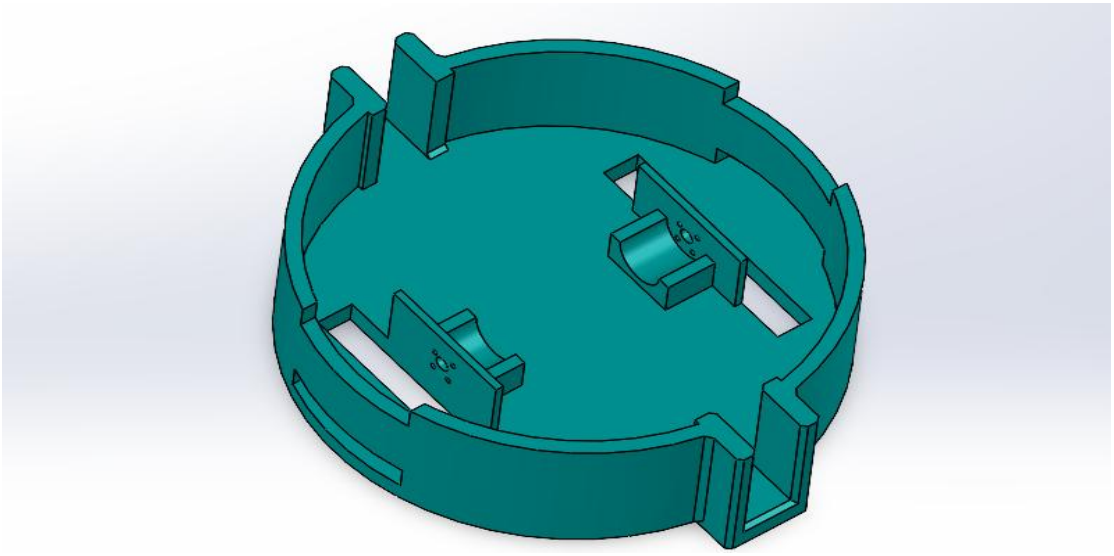
Final Prototype design – Front view



Final Prototype - Isometric



Initial robot draft – Side view



Melty Brain Combat Robot – Internal Section

The final robot design prototype is designated green to highlight it as the officially adopted design. At the same time, the red represents the initial robot's design, although certain components were retained.

Robotic Control

Beyond the more external section of the robot, a crucial and Germane phase in building a Melty brain combat robot is the internal section, otherwise known as the controls. They include electrical components such as motors, wiring, receivers, and, in the case of the autonomous bot, sensors. These components compose the driving system that integrates the mechanical, electrical, and programming segments into the robot, highlighting the need for proper placement of each component within a more concise, consistent, and composed structure. The analysis of electrical schematics, microcontrollers, and mechatronic actuators will effectively describe this aspect of the melty brain combat robot, providing a more detailed perspective on their operation within the bot.

Electrical Schematics

They are diagrammatic, symbolic representations of electrical circuits that use standardized symbols from the Institute of Electrical and Electronics Engineers (IEEE/IEC) and lines to show component connections and logical flow, rather than physical layout. They are essential for designing systems, troubleshooting, and understanding how power reaches loads. Key components include resistors, capacitors, switches, and transistors. Basically, they are the design of what the robotic control elements resemble in a simulated form (JIMBLOM). However, melty brain combat robot has a more compact arrangement of basic components, such as resistors, transistors, and capacitors, into electrical components (motors, ESCs, receivers, etc.) of the robot. Below are vivid descriptions of both the autonomous and non-autonomous schematics.

At the top right of the schematic sheets below in Figure 2.1, the 18V 4S LiPo Battery Pack serves as the primary energy source, in other words, the main bus. The high-current 12 AWG wiring carries raw voltage. This feeds the Main Power Bus (+18V), which directly powers the high-draw components. Sensitive electronics like the Flight Controller (FC) and RC Receiver can't handle 18V. The BEC (Battery Eliminator Circuit) steps this down to a stable 5V/3A to keep the logic chips from frying. The central hub is the MCU Flight Controller (identified as Arduino/ESP32 or STM32F4). In a melty brain, the FC isn't just steering; it’s performing complex trigonometry in real-time. It receives high-rate data from the MPU6050 Accelerometer/Gyro units. These sensors tell the FC exactly how fast the robot is spinning (RPM) and its current angular orientation. By calculating the bot's heading as it spins, the FC pulses the motors at a specific point in the 360-degree rotation. This creates a wobble that moves the robot in a specific direction while it continues to spin like a top.

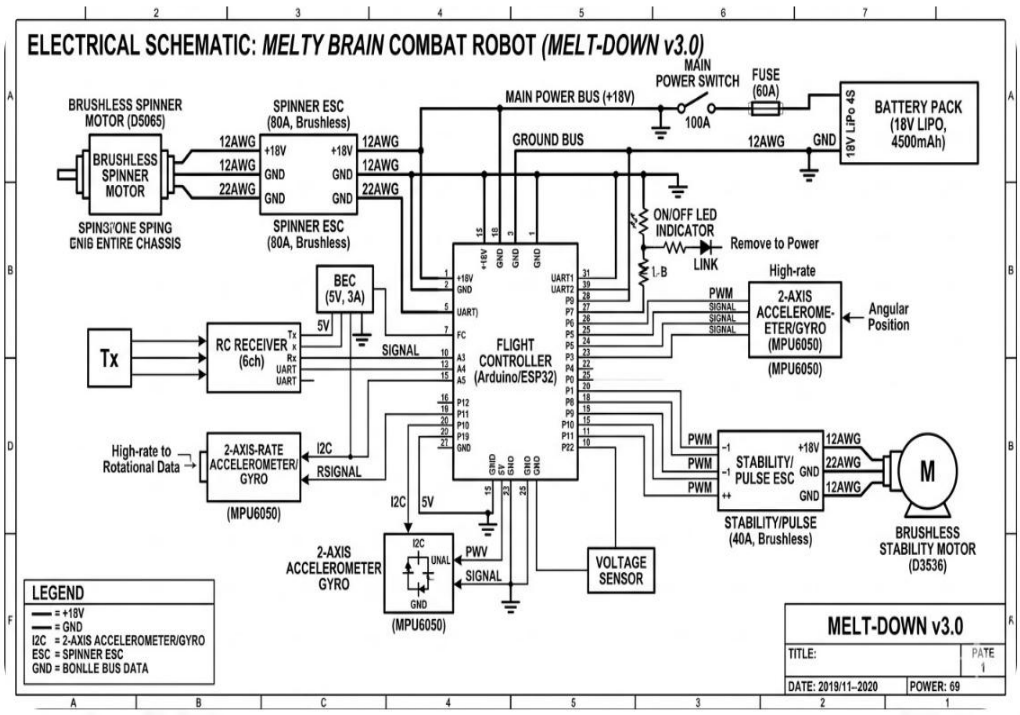


Figure 2.1a – Autonomous Melty Robot Schematics

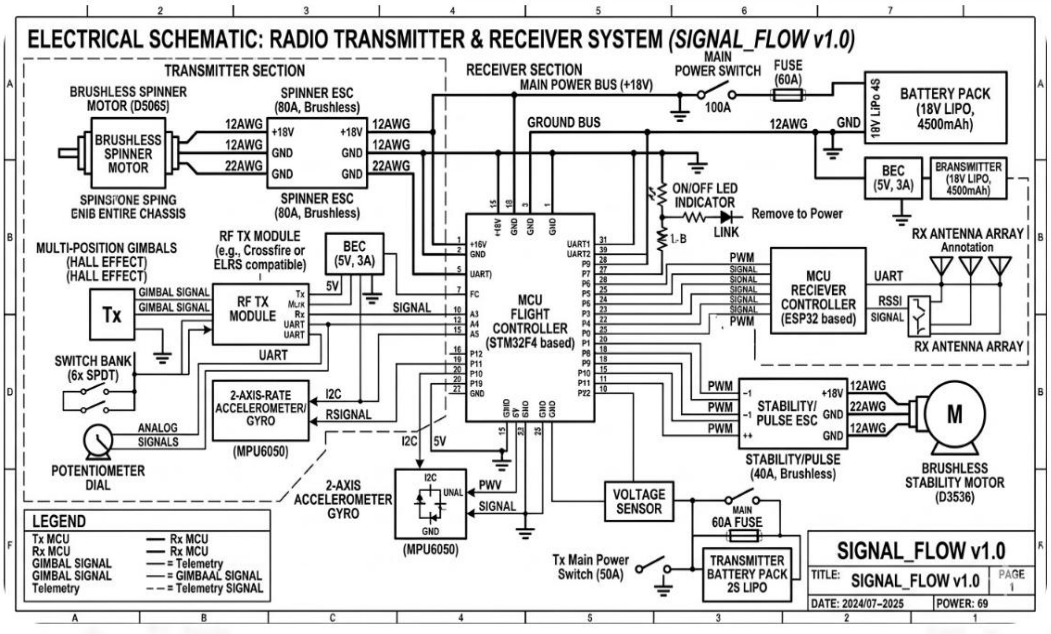


Figure 2.1b – Non-autonomous Melty Brain Schematics

Microcontrollers (MCU) – The Robot's Brain

A microcontroller is a compact, minicomputer on a single integrated circuit (IC) chip designed with the intention to control embedded systems. It incorporates a CPU, memory (RAM/ROM), and programmable input/output (I/O) peripherals to manage specific duties in devices such as household appliances, automotive systems, and Internet of Things (IoT) devices (IBM). Microcontroller platforms that are low-cost, such as Arduino and Adafruit, are preferred by the open-source community for their ease of configuration via IDEs and support for languages like C, C++, and Python.

Microcontrollers, or more advanced versions known as microprocessors (basically computers), are key components in the operation of autonomous and non-autonomous Melty brain combat robots. The absence of a human purview necessitates a properly written programmable microcontroller to accommodate minute changes and diverse scenarios that can occur in a battle arena. These electrical components are critical for determining the timing of drive and weapon deployments. Various microcontrollers recommended for autonomous and automated use include the Raspberry Pi, Adafruit Trinket, Arduino Uno, Teensy, Jetson Nano, and many more. However, this undertaking is scoped down to the Jetson Orin Nano and the Arduino.

The NVIDIA Jetson Orin Nano is a high-performance, energy-efficient AI module engineered for edge computing, robotics, and vision systems. Delivering up to 80 tera operations per second (TOPS) of AI and marking an 80-fold performance evolution over the prior generation's architecture. The platform enables the deployment of complex generative AI models and multi-stage pipelines within a

compact, cost-effective form factor (NVIDIA). Figure 2.2 is an exhibit of a NVIDIA Jetson Orin Nano microcontroller or a computer.

Figure 2.2 – NVIDIA Jetson Orin Nano

The Arduino Uno is a widely used, open-source development board that acts as the brain for electronic projects. Built around the ATmega328P chip (a computer chip that stores and executes your code), it is designed specifically for accessibility. It features 14 digital pins for turning components like LEDs or motors on and off (including 6 pulse width modulation PWM pins for varying power levels) and 6 analog inputs for reading sensor data. Users can easily write and upload code to the board from a computer using the Arduino IDE and a standard USB connection (Arduino). Figure 2.3 shows an Arduino Uno.

Figure 2.3
Arduino Uno Board
With a USB Cable

In non-autonomous mode, there is less emphasis and obligation to use the microcontroller due to human oversight, eliminating the need for cumbersome programming, which occurs only at a minimal or negligible level. Most non-autonomous programs or codes are pre-programmed by the manufacturer to perform certain actions, while users must only set up the interface to their convenience. In non-autonomous systems—such as RF-controlled robots, mobile-controlled robots, or teleoperated machines—the microcontroller receives instructions from a human operator rather than acting independently (Embeddinator). They include a radio transmitter and receiver, Arduino Uno, ESP32, and many more. However, for this robot, the non-autonomous mode adopts a radio transmitter and receiver.

A robotics Transmitter (Tx) and Receiver (Rx) set constitutes a unified wireless link designed for the remote management of robotic platforms. The system operates by converting physical inputs—such as throttle or steering adjustments—into radio waves emitted at a standard 2.4 GHz frequency. Once captured by the onboard receiver, these waves are converted back into electrical impulses. These pulses serve as the primary command signals for the robot's actuators, allowing for the precise, real-time translation of user intent into physical motion (RobotCombatWiki). The one used in this project is from FlySky and uses a process called binding to pair a specific receiver with a specific transmitter, allowing multiple robots to operate in the same area without interference (FlySky). Figure 2.4 is an image of a Flysky transmitter.

Figure 2.4
Radio Transmitter (Tx)
And Receiver (Rx)

Mechatronics - Actuators, Sensor integration, Electromechanical systems

Mechatronics is a combined approach to modern engineering that brings together mechanical systems, electronic control, and computer science. Instead of seeing these fields as separate, mechatronics blends them to create "intelligent" machines. In this system, mechanical parts form the physical structure and provide motion, electronic sensors and circuits serve as the nervous system to collect data, and computer software functions as the brain, processing this data through control theory to make real-time decisions. This comprehensive integration enables the development of highly advanced technologies, such as precision industrial robots, self-driving cars, and complex automated assembly lines used in today's manufacturing (Michigan Tech). The melty-brain combat robot offers a simpler yet vivid example of a mechatronic system. Its 3D-printed TPU-based design forms the mechanical parts, while the electrical wiring, components, and microcontrollers comprise the electrical system. The programming, whether done by hobbyist robot builders or pre-installed by manufacturers, comprises the computer and intelligence system. All this results in a basic blueprint of a mechatronic system. Figure 2.5 illustrates what a simplified mechatronic system looks like in a combat robot.

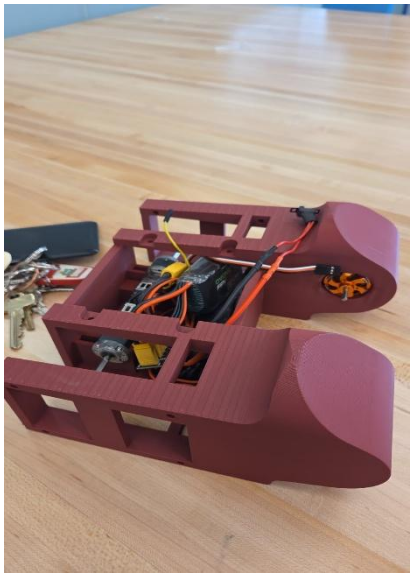


Figure 2.5 – An illustration of a mechatronics system in A regular combat robot (MTSU robotics club, Combat 5, Team robot)

Actuators are the muscles of a robot, responsible for moving and controlling mechanisms by converting energy (electrical, pneumatic, or hydraulic) into physical motion (Roboticsbiz). They enable robots to perform tasks like walking, lifting, and grasping. Common types include electric motors (servos), hydraulic cylinders, and pneumatic actuators (Figelli). The melty rain combat robot uses a brushless 2600kV Outrunner motor as its actuator, powered by a 2S–3S Lipo battery. It has one degree of freedom (1-DOF): rotation about a fixed axis. A fully functional Melty brain has three degrees of freedom for movement: translation along the x- and y-axes (movement across the floor) and rotation (full spin). Figure 2.6 is the pictorial representation of the motor used.

Figure 2.6 – 2600kv Brushless motor

In robotics, sensors are the vital organs that enable a machine to "perceive" its surroundings. They work by detecting physical changes in the environment and translating those changes into electrical impulses that a microcontroller can process. This sensory input is generally categorized into two functional types:

- Proprioceptive Sensors: These monitor the robot's internal state, such as battery levels, joint angles, and motor temperature.

- Exteroceptive Sensors: These interpret the external world, using technologies like LiDAR for 3D mapping, Ultrasonic sensors for distance measurement, and Tactile sensors for "feeling" the grip force on an object.

Ultimately, sensors transition a robot from a blind, pre-programmed machine into an intelligent agent capable of reacting to dynamic, unpredictable environments in real time. The melty brain combat robot uses exteroceptive sensors, specifically a webcam to capture live 3D imaging, a gyrometer to reference location relative to a point and gravity, and an ultraviolet or infrared (UV/IR) sensor to measure distance.

A gyro meter

An IR sensor

A Webcam

Electromechanical systems and components combine electrical engineering and mechanical processes to create motion, control systems, or generate energy. They bridge electrical signals with physical movement, using motors, solenoids, or switches

to operate devices. Common examples include industrial robots, solenoids, relays, and electric motors. The Electronic Speed Controller (ESC) acts as the essential bridge in this system, and, in the context of a combat robot like a melty brain, this relationship is extremely important, as you use electrical pulses to precisely time mechanical rotation at thousands of revolutions per minute (RPM).

Robotic Intelligence

Robotic intelligence represents a fundamental shift from mechanical automation to cognitive autonomy. It is the bridge between a machine that follows a script and one that understands its role within a dynamic world. At its core, robotic intelligence functions through a feedback loop of perception, processing, and physical response. Unlike standard AI—which exists purely in digital environments—robotic intelligence must grapple with the "messiness" of reality, such as friction, gravity, and unpredictable human movement (Shi). Expanding the concept of robotic intelligence to the melty brain (or translational drift) combat robot requires looking at how AI handles the extreme physical paradoxes of this design.

In a melty brain, the entire robot is the weapon. It spins at thousands of RPM while simultaneously "drifting" across the arena. This isn't just movement; it’s a high-speed balancing act of physics that would be impossible without a sophisticated digital "brain" to mediate between the driver’s intent and the machine's violent rotation. For a melty brain, intelligence is not an optional upgrade—it is the enabling technology. Because the robot is constantly spinning, there is no fixed "front." The onboard AI must synthesize data in microseconds to create a "virtual heading." Modern melty brains often use a Kalman filter to merge data from multiple sources—accelerometers for spin, magnetometers for heading, and sometimes IR beacons for absolute positioning. This allows the robot to "reason" through sensor drift, ensuring it doesn't lose track of the opponent's position while spinning at over 3,000 RPM (Beagleboard).

Embedded Vision System

An embedded vision system is a technology that integrates cameras and computer vision software directly into a compact hardware device, enabling it to perceive, interpret, and react to its visual environment in real time. Unlike traditional machine vision, which requires a large, separate computer to process images, embedded vision systems perform processing on the device itself (at the edge), offering low latency, low power consumption, and a small footprint, making them suitable for robotics, automation, and AI applications. Embedded vision is essentially the "brain-eye" integration that allows machines to operate independently of a server room (Sea level). This system efficiently leverages edge computing across every accessory and component, designed to support this concept.

A simple yet rudimentary embedded vision system setup is the Arduino Uno + IR sensor, which is suitable for tasks such as proximity detection, basic motion sensing, or counting, often in low-light conditions. A standard IR (Infrared) setup, such as the FC-51 obstacle avoidance sensor, operates on the principle of Triangulation or Intensity Reflection. The Emitter: An IR LED emits a beam of light at a frequency (usually 38kHz) that is invisible to the human eye but easily detected by silicon sensors. The Receiver: A photodiode or phototransistor waits for that specific frequency to bounce back. It can be viewed as the "distilled essence" of an embedded vision system. While it doesn't process pixels like a camera, it processes photometric data to create a mathematical map of its surroundings. The "Vision" Logic: The Arduino doesn't see "an object"; it sees a voltage change. If the reflected light is strong, the voltage drops (or rises, depending on the module), and the Arduino interprets this as "Object Present" (BoianM).

A more complex but practical embedded vision system setup is the Jetson Orin Nano + Webcam. This setup is ideal for complex AI-driven tasks such as object detection, facial recognition, and autonomous navigation. The USB webcam acts as the eyes, constantly streaming visual data to the Jetson Orin Nano (Irace). The Jetson Nano, running AI algorithms (such as YOLO for object detection), processes video frames locally. Its embedded NVIDIA Ampere GPU acceleration allows for real-time analysis. Upon detecting an object (e.g., a person, an obstacle), the system acts immediately, such as stopping a robot or logging the detection, without sending the video to a cloud server (Dharmalingam).

A melty brain combat robot spins to move using "translational drift," relying on precise timing for directional control. Integrating an Arduino + IR sensor with a Jetson Orin + Webcam creates a high-performance embedded vision system capable of fast-acting opponent detection, accurate orientation tracking, and autonomous translational steering (Halfacree). The System Architecture includes a Jetson Orin (Nano/NX/AGX) that functions as the high-level brain, handling image processing, object detection (using AI/computer vision), and strategy, a Webcam (USB or MIPI-CSI) that is mounted to look forward (or omnidirectionally), providing visual input, or a Arduino (Microcontroller) that acts as the low-level controller handling real-time sensor data, motor driver timing (essential for melty brain), and communicating and a IR Sensor (TCRT5000 or similar) that provides proximity sensing or orientation feedback by detecting a "beacon" or the arena wall.

The Advantages of this approach include, but are not limited to, High-Speed processing, which involves Jetson that Orin provides enough TOPS (Trillions of Operations Per Second) to run advanced AI models at high FPS, essential for fast-spinning bots, and real-time responsiveness that occurs by splitting the workload. At

the same time, Arduino handles high-speed motor timing; Jetson handles complex vision; and the system avoids latency issues. Lastly, robustness is achieved through IR sensors that can be modulated to avoid interference from arena lighting, providing a reliable trigger mechanism even when the visual camera is obscured or blinded (Spencer).

Autonomous Mode Operations

This highlights the programming and code used to automate the robot's response to sensor data. This ensures the robot has direction and can properly and efficiently react to changes in the Arena. Robotic programming is the process of encoding software instructions that enable robots to perceive their environment, use data perceived to make decisions, and perform specific tasks autonomously. It involves designing, testing, and implementing code—ranging from simple motion sequences to complex AI algorithms—using methods like offline simulation or direct "teach-pendant" guidance (MATLAB). Robotic programming is the "soul" of the machine. It is the bridge between inanimate hardware (motors and metal) and purposeful behaviour. Robotic programming occurs at different levels of abstraction, depending on the goal: from low-level languages such as C/C++, mid-level languages such as Python/ROS, and finally, high-level languages such as VLA models. This melt-brain combat robot used the Arduino IDE, whose syntax is derived from C++, to program the Arduino Uno, and a Python script to program the Jetson Orin Nano.

The program for the robot is attached below, with comments that clearly highlight the role each section plays in automating the robot.

Arduino Code/Programming

```
//================================================================
=====

// Melty Brain Combat Robot — Bidirectional Drive Only

// Drive:   2× Brushless motors via bidirectional ESCs

// Weapon:  The robot body IS the weapon — full spin attack

// Sensor:  Sharp GP2Y0A21YK0F analog IR (10–80 cm)

// Pin map:

//   D4  → Left drive ESC signal

//   D5  → Right drive ESC signal

//   A0  → IR sensor output

//   GND → Both ESC grounds + Arduino GND (COMMON GROUND
REQUIRED)

// Bidirectional ESC throttle:

//   1000 µs = full reverse

//   1500 µs = neutral / stop
```

```
//    2000 µs = full forward

//============================================================

#include <Servo.h>

// ── ESC signal pins
// ──────────────────────────────────────────────────────────

#define ESC_LEFT_PIN   4

#define ESC_RIGHT_PIN  5

// ── IR sensor pin
// ──────────────────────────────────────────────────────────────

#define IR_PIN         A0

// ── ESC throttle microseconds
// ──────────────────────────────────────────────

const int ESC_NEUTRAL      = 1500;   // Stop

const int ESC_SLOW_FWD     = 1600;   // Slow forward  (scanning spin)

const int ESC_SLOW_REV     = 1400;   // Slow reverse  (scanning spin)

const int ESC_CHARGE_FWD   = 1800;   // Fast forward  (charge)

const int ESC_ATTACK_FWD   = 1800;   // Full forward  (attack spin)

const int ESC_ATTACK_REV   = 1000;   // Full reverse  (attack spin)

// ── Distance thresholds (cm)
// ──────────────────────────────────────────────

const float DETECT_DISTANCE_CM = 30.0;   // Begin charge
```

```
const float STOP_DISTANCE_CM   =  3.8;   // ~1.5 in → trigger attack

// ── Attack duration
──────────────────────────────────────────────────────────

const unsigned long ATTACK_DURATION_MS = 3000;

// ── IR calibration
──────────────────────────────────────────────────────────

const float IR_K      = 4800.0;

const int   IR_OFFSET = 20;

const float IR_MIN_CM =  2.0;

const float IR_MAX_CM = 80.0;

// ── State machine
──────────────────────────────────────────────────────────

enum RobotState { SCANNING, CHARGING, ATTACKING };

RobotState currentState = SCANNING;

unsigned long attackStart = 0;

Servo escLeft;

Servo escRight;

//──────────────────────────────────────────────────────────

//  IR distance

──────────────────────────────────────────────────────────
```

```
float readDistanceCM() {
  int raw = analogRead(IR_PIN);
  if (raw <= IR_OFFSET) return IR_MIN_CM;
  float d = IR_K / (float)(raw - IR_OFFSET);
  d = constrain(d, IR_MIN_CM, IR_MAX_CM);
  return d;
}
//──────────────────────────────────────────────────────────────
//  Drive helpers
//──────────────────────────────────────────────────────────────
// Full stop — neutral on both ESCs
void stopDrive() {
  escLeft.writeMicroseconds(ESC_NEUTRAL);
  escRight.writeMicroseconds(ESC_NEUTRAL);
}
// Slow spin in place for scanning
// Left forward + right reverse = clockwise body rotation
void slowSpin() {
  escLeft.writeMicroseconds(ESC_SLOW_FWD);
  escRight.writeMicroseconds(ESC_SLOW_REV);
```

```
}

// Both motors forward to close the distance on target

void chargeForward() {

  escLeft.writeMicroseconds(ESC_CHARGE_FWD);

  escRight.writeMicroseconds(ESC_CHARGE_FWD);

}

// Full speed opposite directions — maximum RPM spin attack

// Left full forward + right full reverse = full body revolution

void attackSpin() {

  escLeft.writeMicroseconds(ESC_ATTACK_FWD);

  escRight.writeMicroseconds(ESC_ATTACK_REV);

}

//──────────────────────────────────────────────────────────────────────

//  ESC arming

//  Bidirectional ESCs need to see neutral (1500 µs) on startup

//  to calibrate their center point before accepting commands.

//──────────────────────────────────────────────────────────────────────

void armESCs() {

  Serial.println("Arming ESCs — holding neutral for 3 s…");

  escLeft.writeMicroseconds(ESC_NEUTRAL);
```

```
  escRight.writeMicroseconds(ESC_NEUTRAL);
  delay(3000);
  Serial.println("ESCs armed.");
}
//──────────────────────────────────────────────────────────────────────
//  Setup
//──────────────────────────────────────────────────────────────────────
void setup() {
  Serial.begin(9600);
  escLeft.attach (ESC_LEFT_PIN,  1000, 2000);
  escRight.attach(ESC_RIGHT_PIN, 1000, 2000);
  armESCs();
  currentState = SCANNING;
  Serial.println("Melty Brain — SCANNING");
}


//──────────────────────────────────────────────────────────────────────
//  Main loop
//──────────────────────────────────────────────────────────────────────
void loop() {
```

```
float dist = readDistanceCM();
Serial.print("Dist: ");
Serial.print(dist, 1);
Serial.print(" cm | State: ");
switch (currentState) {
  // ── SCANNING
```

```
  case SCANNING:
    Serial.println("SCANNING");
    slowSpin();
    if (dist <= DETECT_DISTANCE_CM) {
      stopDrive();
      delay(50);
      Serial.println(">>> Target detected — CHARGING");
      currentState = CHARGING;
    }
    break;
  // ── CHARGING
```

```
  case CHARGING:
```

```
  Serial.println("CHARGING");
  chargeForward();
  if (dist <= STOP_DISTANCE_CM) {
    stopDrive();
    delay(80);
    attackStart  = millis();
    currentState = ATTACKING;
    Serial.println(">>> In range — ATTACKING");
  } else if (dist > DETECT_DISTANCE_CM) {
    stopDrive();
    currentState = SCANNING;
    Serial.println(">>> Target lost — SCANNING");
  }
  break;

// ── ATTACKING
// ─────────────────────────────────────────────────────────────
case ATTACKING:
  Serial.println("ATTACKING");
  attackSpin();   // Left full fwd, right full rev = full revolution
```

```
      if (millis() - attackStart >= ATTACK_DURATION_MS) {
        stopDrive();
        delay(300);   // Let motors spin down before next state
        currentState = SCANNING;
        Serial.println(">>> Attack done — SCANNING");
      }
      break;
    }
   delay(20);
}
```

Arduino File: melty_brain_robot.ino

Unique Insight into the Adoption and Ethics of Autonomous Combat Robots

A key factor in this research is the introduction of autonomous combat robots into the battle Bot circuit. Although it is important to note that this has been tried in more professional circuits like the popular BattleBots, it is used much more limitedly, even at a much more advanced level. In engineering, the adoption of AI is increasing at an alarming rate, and it is clear to every robot hobbyist and enthusiast that autonomous and automated robots are the future of robotics and are here to stay. Therefore, every member of the robotic world can only follow the evolving trend but must do so within the standards and ethics of engineering. This must be accomplished by keeping the robots' standards aligned with the Hierarchy of control protocols.

The initial protocol is to eliminate or replace hazardous materials or processes with safer alternatives. This can be achieved by ensuring the robot follows strict, specific pseudocode for its actions in the arena, while also implementing a shutdown or failsafe that automatically terminates all operations across the mechanical, electrical, and intelligence systems.

A lock is placed on weapons to ensure they are safe when not in use, and the robot is not dangerous or harmful to its environment, aligning with the second protocol of engineering controls, which aims to prevent hazards by locking or making certain areas inaccessible that could be dangerous for human interaction.

Every potential robot controller or user must be made aware of what each component does, and, if not possible, at least familiar with the components germane to the robot's functionality. This is in line with the administrative control protocol, under which users with access to robots and their environments are aware of and kept up to date on the robot’s standard operating procedures. Individuals not given

permission or informed about the robot protocol and usage are to be forbidden or strictly monitored when near the robot.

Lastly, a key safety protocol is that users use protective equipment (PPE). This is, in fact, the most important safety protocol for every individual associated with the robot and its environment, as it can help ensure minimal or negligible damage to people if the above three components are not adhered to. For the battle arena, safety glasses are highly recommended, as robots often have pieces flying around from collisions. Shoes that properly cover toes and pants without slits or tears.

Featured/Possible Improvements

This research adopted a simpler, rudimentary intelligence approach using the Arduino Uno and IR sensors to demonstrate proof of concept within the given timing and workforce constraints. Still, extensive research was conducted on the more advanced Jetson Orin Nano + Webcam approach, with practical use demonstrated through pseudocode, programming, and code execution.

This research is but a simplification of many possibilities for the robot's autonomous operation. AI agents and large language models (LLMs) were among the many possibilities for integrating into a more advanced, specialized robotic control system, as they enable robots to learn as they operate by cross-referencing previous data in real time. The Kalman filter and virtual heading, briefly mentioned earlier, are good applications of tools that can be adopted in a melty brain combat robot to improve performance. Given adequate time, knowledge, and resources, a more extensive, faster, and efficient model can be designed and implemented in the combat robot’s circuit.

Test and Execution

The Melty Brain robot competed in the National Robotics Challenge in Marion, Ohio. Throughout the tournament, the robot effectively performed the core functions required for combat-level competition. However, significant challenges within the drive system were observed and partially mitigated, though these issues reduced the robot's overall capacity by over 40%.

A primary point of failure was the axle system, which secured the motors to the wheels. Because the tires controlled both the drive and weapon systems, high-torque motors were required; however, the axle system was not sufficiently balanced to handle this power. This imbalance resulted in accelerated tire wear, far exceeding that of a typical combat robot. Despite these setbacks, a functional prototype was successfully demonstrated, proving the design’s potential for future combat readiness.

Below is video evidence of the robot at the National Robotics Challenge (NRC).

https://youtube.com/shorts/x723LXu1xjs?si=DoRz53lz6UaCSoyG

Bill of Materials

S/N	Material	Amount($)	Total($)
4	Motors	19.99	79.96
2	Accelerometer	17.1	34.2
1	Adupro Robot Controller	33	33
4	Weapon ESC	26.89	107.56
2	Arduino Microcontroller	16.99	33.98
1	XT30 Connectors	12.99	12.99
2	PD board	10	20
1	Hexagonal M3 spacers	7.92	7.92
1	M3 head screws	4.99	4.99
4	Lipo Battery	18	72
1	Jetson Nano computers	249	249
1	Rollers	8.99	8.99
1	XT30 connectors	12.99	12.99
4	PLA Filament	19.99	79.96
3	TPU Filament	38.99	116.97
4	Rubber Tires	15.99	63.96
4	Aluminium Couplings	7.69	30.76
	Total		968.83

A total budget of $1,000 was approved for this research, with surplus spent on tax deductions for components acquired.

Key Takeaways

This innovative inquiry on the Melty Brain combat robot gave a unique insight into the application of autonomous and non-autonomous driven systems in combat robotics, delving into the comparisons and juxtapositions of the machinations, risks, ethics, and protocols to establish a platform and a means for safe, precise, and well-guided integration of AI (a key component of an autonomous system) in combat robotics and robotics in general for further robotic innovations and research purposes.

The objectives of building an autonomous and non-autonomous combat robot were met, and the integration of an autonomous system, while highlighting both rudimentary and advanced systems, was implemented during this experimental and innovative inquiry.

References

- Melty Brain Combat Robot Circuit, beagleboard.org. https://share.google/lkQBoW6Vy67WTVmzu
- Robotics Design Process, Galileo Educational Network. https://www.galileo.org/robotics/design.html
- (2026) contest manual, National Robotics Challenge. https://www.thenrc.org/contest manual
- (2022, June 9th). TPU Filament – The Good, The Bad, And the Ugly. MASSIVIT. https://www.massivit3d.com/blog/tpu-filament-the-good-the-bad-and-the-ugly/#:~:text=High%20elasticity,Weather%20resistance
- https://ijirem.org/view_abstract.php?title=Enhancing-Robotic-Arm-Performance:-Integrating-Arduino-Control-and-Aerodynamic-Principles-for-6-Degrees-of-Freedom&year=2023&vol=10&primary=QVJULTE2MTc=
- JIMBLOM, https://learn.sparkfun.com/tutorials/how-to-read-a-schematic/all
- https://www.ibm.com/think/topics/microcontroller#:~:text=A%20microcontroller%20unit%20(MCU)%20is%20a%20small,communications%20*%20Home%20appliances%20*%20IoT%20devices
- NVIDIA, https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/nano-super-developer-kit/
- Arduino, https://docs.arduino.cc/hardware/uno-rev3/
- Embeddinator, https://www.embeddinator.com/product/rf-wireless-controlled-non-programmable-robotic-diy-kit-without-microcontroller#:~:text=Description-,RF%20Wireless%20Controlled%20Non%2DProgrammable%20Robotic%20DIY%20Kit%20(Without%20Microcontroller,screw%20driver%2C%20etc...

- RobotCombatWiki, https://robotcombatwiki.com/wiki/Radios#:~:text=From%20RobotCombatWiki,bots%20and%20other%20RC%20devices.
- FlySky, https://www.flysky-cn.com/
- Michigan Tech, https://www.mtu.edu/applied-computing/undergraduate/mechatronics/what-is/#:~:text=Innovation%20and%20Automation:%20Mechatronics%20has,more%20capable%20machines%20and%20products.
- Firgelli, https://www.firgelliauto.com/pages/news-actuators
- Zhongzhi Shi, https://www.sciencedirect.com/topics/computer-science/intelligent-robots#:~:text=A%20robot%20is%20a%20programmable,is%20emerging%20called%20the%20agribot.
- Beagleboard.org, https://www.beagleboard.org/projects/melty-brain-combat-robot-circuit#:~:text=While%20most%20combat%20robots%20have,into%20producing%20rotational%20kinetic%20energy.
- Sealevel, https://www.sealevel.com/embedded-vision-machine-vision#:~:text=Embedded%20vision%20systems%20combine%20all,stringent%20SWAP%2DC2%20requirements.
- Boian M, https://www.instructables.com/Arduino-Nano-Infrared-Obstacle-Avoidance-Sensor-Wi/
- Dharmalingam, https://forums.developer.nvidia.com/t/cameras-for-nvidia-jetson-orin-nano-development-kits/262648

- Halfacre, https://www.hackster.io/news/open-melt-is-a-melty-brain-translational-drift-robot-spinning-like-a-remote-control-beyblade-8090c801724e#:~:text=%22A%20translational%20drift%20robot%20spins%20its%20entire,brain.%22%20%22To%20achieve%20this%2C%20the%20rate%20of
- Github, https://github.com/nothinglabs/openmelt2/
- Spencer, https://www.swallenhardware.io/battlebots/2018/4/29/halo-pt2-meltybrain#:~:text=If%20you%20put%20a%20light,processor%20much%20time%20to%20th
- https://www.mdpi.com/1424-8220/25/10/3126#:~:text=This%20frame%20rate%20enables%20the,thermal%20images%20from%20two%20sensors.
- MATLAB, https://www.mathworks.com/discovery/robot-programming.html
- Jha, S. K., Tarafder, M., Dhibar, D., Pokharel, B., Shrestha, J., & Koirala, K. B. (2020). Design and Development of a Soil Moisture-Based Automatic Irrigation System in Nepal. https://doi.org/10.37899/journallamultiapp.v1i3.197
- Brian Irace, Why develop a native mobile application? https://irace.me/native
- Linux hint: Does Arduino support ESP32-C3? https://linuxhint.com/arduino-support-esp32-c3/
- Fiberoptic | Hackaday. https://hackaday.com/tag/fiberoptic/
- Roboticsbiz, What are the different types of robot control systems? https://roboticsbiz.com/what-are-the-different-types-of-robot-control-systems/
- (2025) *Intuitive Da Vinci: World-class Robotic Surgical Systems,* INTUITIVE. https://www.intuitive.com/en-us/products-and-services/da-vinci

- Rich Olson (2025). *Melty Brain*, Robot Combat Wiki. https://robotcombatwiki.com/wiki/Melty_Brain#:~:text=The%20Melty%20Brain%20robot%20is,e.g.%20to%20escape%20from%20corners.&text=Your%20browser%20can't%20play%20this%20video.&text=An%20error%20occurred.,is%20disabled%20in%20your%20browser.
- Stern Alexander (2025). *From Sci-fi to Reality: Robotics in Everyday Life,* Medium. https://medium.com/@sternalexander/from-sci-fi-to-reality-robotics-in-everyday-life-1c1ddc0cd707
- Gentry Clark (2018). *Melty Brain Combat Robot Circuit,* Hackster.io. Melty Brain https://share.google/Vc5kjsJTOvcOlO83M
- Wikipedia, https://en.wikipedia.org/wiki/Degrees_of_freedom_(mechanics)#:~:text=The%20degree%20of%20freedom%20of,the%20other%20five%20are%20specified.