NAIT

Edmonton, Alberta

Automatic Connect Four

As a submission to

Mr. AJ Armstrong, Instructor

&

Gary Munro, Instructor

English and Communications Department

As a submission by

Adrian Baira, Student

&

Naresh Koirala, Student

CMPE 2965

April 14, 2025

Statement of Contributions

This document was co-authored by Adrian Baira and Naresh Koirala. Both participants reviewed the contributions outlined in the table for accuracy, and we determined that the division of effort is roughly equal.

|  |  |  |
| --- | --- | --- |
| Section | Author | Reviewer |
| Title Page, Table of Contents, Cover Letter, List of Figures and Tables, and general Formatting | Adrian Baira & Naresh Koirala | Adrian Baira & Naresh Koirala |
| Abstract | Adrian Baira & Naresh Koirala | Adrian Baira & Naresh Koirala |
| Introduction | Adrian Baira & Naresh Koirala | Adrian Baira & Naresh Koirala |
| Overview | Adrian Baira & Naresh Koirala | Adrian Baira & Naresh Koirala |
| Design | Adrian Baira & Naresh Koirala | Adrian Baira & Naresh Koirala |
| Implementation | Adrian Baira & Naresh Koirala | Adrian Baira & Naresh Koirala |
| Project Results | Adrian Baira & Naresh Koirala | Adrian Baira & Naresh Koirala |
| Conclusion | Adrian Baira & Naresh Koirala | Adrian Baira & Naresh Koirala |
| References | Adrian Baira & Naresh Koirala | Adrian Baira & Naresh Koirala |

Northern Alberta Institute of Technology

11762 - 106 Street NW

Edmonton, AB, Canada

T5G 2R1

March 12, 2025

Mr. AJ Armstrong, Mr. Gary Munro

Instructors

Northern Alberta Institute of Technology

11762 - 106 Street NW

T5G 2R1

Dear Mr. Armstrong and Mr. Munro:

As per the requirements of CMPE2965, we are submitting the report titled “Automatic Connect Four” for evaluation. This project, which aims to develop an automated, online version of the classic Connect 4 game that repurposes old game boards for digital integration, is a testament to our innovative spirit. Players can compete against a computer or another user. The purpose is to combine entertainment with skill-building through strategic gameplay while giving new life to existing resources.

This report details the overall design and implementation of the Connect 4 project. Further outline are the successes and roadblocks encountered throughout this project. This report details the electronics used and the programming structure utilized in this project.

We want to thank our instructors throughout our NAIT career. Their patience and enthusiasm in educating theory students like us have been instrumental in our learning journey. Their genuine interest in our success has fostered a strong desire to learn and excel.

Sincerely,



Adrian Baira and Naresh Koirala

CNT Students

Table Of Contents

Contents

[List Of Figures 5](#_Toc195562690)

[1.0 Introduction 7](#_Toc195562691)

[2.0 Overview 8](#_Toc195562692)

[3.0 Design 9](#_Toc195562693)

[3.0.1 Research 9](#_Toc195562694)

[3.1 3D-Design 11](#_Toc195562695)

[3.1.1 Case Design 11](#_Toc195562696)

[3.1.2 Electronics Design 14](#_Toc195562697)

[3.1.3 Solenoid Circuit Design 16](#_Toc195562698)

[3.2 Software Development 17](#_Toc195562699)

[3.2.1 Web Interface Development 17](#_Toc195562700)

[3.2.2 Client-side Development 18](#_Toc195562701)

[3.2.3 Game Logic Development 19](#_Toc195562702)

[3.3 Physical Component 20](#_Toc195562703)

[3.3.1 Pico W 20](#_Toc195562704)

[3.3.2 ESP-32 21](#_Toc195562705)

[3.3.3 Stepper Motor Driver 21](#_Toc195562706)

[3.3.4 SENSOR THRU-BEAM 508MM OPEN COL 22](#_Toc195562707)

[4.0 Implementation 23](#_Toc195562708)

[4.1 Hardware and Mechanical Assembly 23](#_Toc195562709)

[4.2 Software Integration 24](#_Toc195562710)

[4.3 Testing and Refinement 24](#_Toc195562711)

[5.0 Project Results 25](#_Toc195562712)

[6.0 Conclusion 25](#_Toc195562713)

[7.0 References 26](#_Toc195562714)

# List Of Figures

[Figure 1 - Prototype Sensor Holder 13](#_Toc195562715)

[Figure 2 Token dropper holder 14](#_Toc195562716)

[Figure 3 Token Holder 15](#_Toc195562717)

[Figure 4 Wiring Diagram 16](#_Toc195562718)

[Figure 5 Initial Solenoid Circuit 17](#_Toc195562719)

[Figure 6 Final Solenoid Circuit 18](#_Toc195562720)

[Figure 7 Website Home Page 18](#_Toc195562721)

[Figure 8 During the Game 19](#_Toc195562722)

[Figure 9 Initial Game Logic Code 20](#_Toc195562723)

[Figure 10 Pico W 21](#_Toc195562724)

[Figure 11 ESP-32 22](#_Toc195562725)

[Figure 12 TMC 2209 23](#_Toc195562726)

[Figure 13 5 mm IR break-beam sensor pair for token detection (Adafruit, 2018). 23](#_Toc195562727)

Abstract

The Automated Connect 4 project was created to modernize the automated game version while maintaining simplicity and improving interactivity. The primary goal of this project was to provide an engaging user experience by automating gameplay, such as token placement and tracking of tokens, by integrating hardware and software components. This project mainly uses the ESP-32, which controls the game logic, motor control, and web server elements, and the Raspberry Pi Pico W, which runs sensor inputs and communicates with the ESP-32 via the MQTT protocol.

This project's results were measured by testing the synchronization of components during gameplay and assessing their responsiveness. Token placement accuracy needed to be tested by moving the token holder from one side to another. The sensor was tested by repeating token drop, which showed that changes in these components produced visual effects on token tracking and game progress, satisfying the goal of automating an old gameboard. Additionally, the goal of developing a system was achieved by operating the game board through the web interface and viewing the updated gameplay and interactions. These tests demonstrated that all project elements, such as hardware, software, 3D design and user interface, were successfully integrated and worked together cohesively as a single unit.

# 1.0 Introduction

Automation has become an increasingly vital application of engineering technology, whether it be in industrial or commercial settings. The Automated Connect 4 project incorporates these automation and digital systems to transform the classic Connect 4 board game into an interactive and engaging experience. This project repurposes an existing automatic Connect 4 game board and incorporates hardware and software components to create an automated gaming experience for solo and two-player modes.

This report aims to document the development process of this automated Connect 4 system, including the design, implementation, and results achieved throughout this project. The report also addresses key challenges encountered in completing the automatic connect four, such as hardware compatibility and system synchronization, and determining the innovative solutions to overcome these challenges, like using MQTT for Pico W and ESP32 communication (main.py). The research included studying sensors from Adafruit Industries(2025), motors from Adafruit Industries(2024), and Micro Python docs for coding the game logic and web interface (game\_logic.py, website.py). This report concerns instructors and automation students interested in IoT projects (Baira & Koirala, 2025).

The primary objective of this project is to automate gameplay interactions, including disc placement and tracking, while improving user attention through an intuitive web interface (script.js). The project's success will be assessed based on its ability to fulfill these goals while delivering a distinctive, strategic gaming experience demonstrating modern automation technologies' potential.

# 2.0 Overview

The Automatic Connect 4 project transforms the traditional Connect 4 game into an automated, technology-enhanced experience by integrating hardware and software to make a different gaming experience. The main goals were to automate token placement and tracking, repurpose an existing game board for digital functionality, and create an engaging player experience through a web interface, supporting two-player and solo modes against an AI opponent.

This project uses two key microcontrollers: the Raspberry Pi Pico W, which handles sensor inputs to detect token positions, and the ESP32, which manages game logic and motor control and hosts a web server for user interaction. A Laser sensor tracks token drops, while a stepper motor and solenoid system automate token placement. The MQTT protocol enables real-time communication between the microcontrollers, ensuring synchronized gameplay. A belt-driven rail system that moves the token holder across the board and a custom cylinder with ramp stores and releases tokens efficiently.

Implementation involved assembling hardware components, programming the microcontrollers, and developing a web interface. Challenges include ensuring reliable MQTT communication, synchronizing the microcontrollers, stabilizing the rail system, and token placement through iterative testing and refinement. The final system achieved a 97% sensor detection accuracy and motor precision within 1mm, successfully repurposing the original game board while adding automation parts. The result is an interactive automatic Connect 4 experience that combines classic gameplay with modern technology, with ongoing work to enhance the AI bot for varied difficulty levels.

# 3.0 Design

This section outlines the design process of the Automatic Connect Four system, covering research, 3D designs, software development, and physical components. It details the selection of motors and sensors, the creation of 3D-printed parts for token handling, the development of the web interface and game logic, and the integration of hardware like the Pico W and ESP32 (main.py, tmc2209.py). Each subsection addresses key challenges and solutions to ensure precise automation and an engaging user experience (Baira & Koirala, 2025).

## 3.0.1 Research

This project's initial step started with searching for a motor to ensure precise token drop movement. This aligns with this project's goal of achieving precise movement. There are several motor options, with the minimum requirement being the ability to move forward and reverse. After thoroughly exploring different motors, a 12V DC motor and a 12V stepper motor were selected. A DC motor is simple to control, has variable speed, less precision control, and is inexpensive. (RayMing PCB, 2024).

On the other hand, a stepper motor is hard to control regarding speed, but it is known for its precise positioning. However, the cons are that it is challenging to maintain (RayMing PCB, 2024). Although the stepper motor is lacking in some areas compared to a DC motor, it was a clear sign that it was the perfect fit for this project. Its precise motor control capabilities would ensure that it would move a token into the slot precisely every time, instilling confidence in this project's direction. 

This project's next challenge was determining the sensor that best fit this project. It needed to be something that would not interfere with the gameboard, and there were many different types of sensors to select. Several choices were considered, including pressure, impact, and laser break sensors. Pressure sensors detect force or weight changes, and they are placed beneath each column to register a token's landing, offering durability for consistent move detection. However, putting them across the bottom of the grid could increase wiring complexity, and they might be sensitive to vibrations or uneven token placement (UpKeep, 2023). In addition, to reset the gameboard for another game, the pressure sensors would have to be removed before each game, causing potential user error, which would impact the reliability of this project. The other option was impact sensors, which often use piezoelectric technology to detect sudden vibrations. A token would hit a column's base or stack to detect a placement; these sensors provide cost. However, the impact sensor flaw in this project is the potential false triggers from external shocks, such as table movements or sudden bumps, requiring careful tuning to maintain reliability (GlobalSpec, 2025). It would also need to be removed after each game. Laser break sensors use a beam to detect an object. For example, if a token passes through, It will trigger a voltage to signal that something has broken the laser, which provides high accuracy and fast response for real-time tracking. These sensors can combine seamlessly with the board's design but need precise alignment and could be affected by dust or light interference. (KJT Sensors, 2025). Their design provides minimal interference with the original board's design. After more research, laser break sensors were chosen for their excellent accuracy, fast detection, and non-intrusive design with the gameboard, ensuring accurate tracking of player moves while preserving the original design of the gameboard.

This project used a Raspberry Pi Pico W and an ESP32 to integrate the components for their complementary strengths in creating an automated Connect 4 system. The Pico W was chosen for its cost-effectiveness, sufficient GPIO pins for component interfacing, and integrated Wi-Fi for reliable MQTT communication, supporting efficient data exchange (Raspberry Pi, 2022). The ESP32 was selected for its powerful dual-core processor, robust Wi-Fi capabilities to host a web server for player interaction with the game and AI opponent, and ample GPIO to manage the 12V stepper motor and two solenoids one for releasing tokens into columns and one for dispensing a token to the token holder (Espressif Systems, 2023). MQTT was a lightweight messaging protocol to simplify real-time communication between the microcontrollers, ensuring smooth coordination of game logic, token detection, and mechanical actions (MQTT, 2024). This research gave the project a robust system for an automated Connect 4 board experience.

## 3.1 3D-Design

### 3.1.1 Case Design

The first hurdle during the design process was figuring out how to mount the sensors in such a way as to avoid messy wiring and not disrupt the original game board. At first, the aim was to mount the sensors onto the walls of the game board. However, the problem with mounting it that way is that the width of the sensor would interfere with game piece placement (see Figure 1). After consulting and brainstorming, a frame was created to rest atop the board; Figure 1 shows the initial prototype of the frame.

Figure 1 - Prototype Sensor Holder

A black rectangular object with holes

AI-generated content may be incorrect.

This solution alleviates wiring complications and does not change the original game board. Once the sensor case design was completed, A designed a simple holder with a solenoid to hold a token and prevent it from dropping until it was time to drop it (As shown in Figure 2).

Figure 2 Token dropper holder

A black box with a lid open

AI-generated content may be incorrect.

Attention shifted to developing a motor and rail system. The motor and rail system was adapted from a model demonstrated by (Hacker Twins, 2023, May 23) in designing a belt-driven rail system with a stepper motor. For this project, the middle component of their design was replaced with a custom mechanism to hold a token and a solenoid instead of a servo motor. The rotation of the motor drives a gear with a belt across the board so that it moves the token holder left and right across the game board, enabling seamless gameplay.

Lastly, this project needed a token holder to hold all the pieces for the game. Initially, a hopper with a large opening was considered to hold all the pieces, but after some thought, the potential of the tokens was jamming due to the wide opening. A better solution is to use a cylinder design instead so that the user can organize pieces and remove any risk of the tokens being stuck. This decision was made after carefully considering the potential issues with the hopper design; the cylinder design provides a more user-friendly and reliable solution. A ramp was added to the bottom of the dispenser to guide the token accurately to the token holder. (as shown in figure 3)

Figure 3 Token Holder

A 3d model of a cylinder


### 3.1.2 Electronics Design

The system connects several components for the automatic Connect 4 board. The Raspberry Pi Pico W connects seven laser break sensors, each wired to a digital GPIO pin with a 10kΩ pull-up resistor to maintain a 5V high signal. When a Connect 4 token interrupts a sensor’s laser beam, the signal drops, detecting the token’s position in the game grid. The ESP32 controls a stepper motor via PWM pins through a motor driver, using a 12V power supply with a stepper motor driver; the motor driver is used to adjust the motor’s direction and speed for precise movement of dropping the token. Additionally, two ESP32 GPIO pins manage a 12V solenoid, driven through a transistor circuit, to perform actions like releasing discs or resetting the board. Flyback diodes are placed across the solenoid terminals to protect against voltage spikes.

Figure 4 Wiring Diagram

A diagram of a circuit board

AI-generated content may be incorrect.

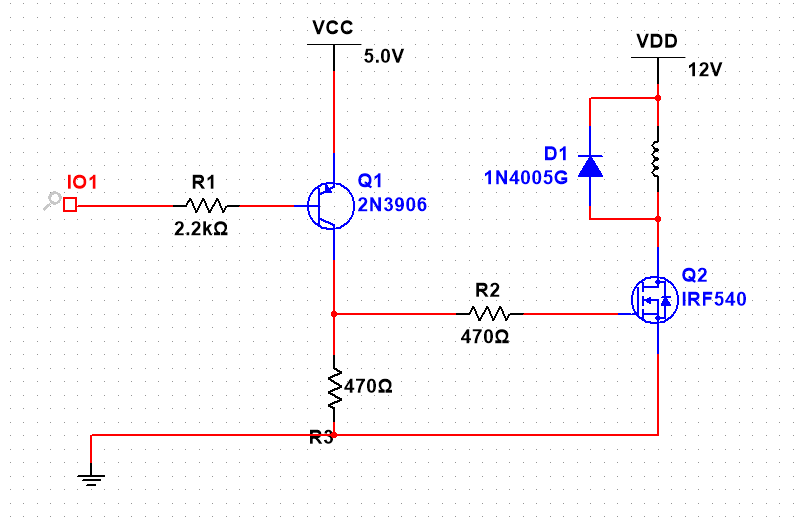
Together, these components allow automated gameplay by detecting disc placements and performing mechanical actions (See Table 1) for all connections to GIPO connections to each microcontroller.

Table 1

|  |  |  |
| --- | --- | --- |
| Device | Microcontroller | Pin Number |
| Laser Break Sensor 1 | Pico W | GPIO22 |
| Laser Break Sensor 2 | Pico W | GPIO21 |
| Laser Break Sensor 3 | Pico W | GPIO20 |
| Laser Break Sensor 4 | Pico W | GPIO19 |
| Laser Break Sensor 5 | Pico W | GPIO18 |
| Laser Break Sensor 6 | Pico W | GPIO17 |
| Laser Break Sensor 7 | Pico W | GPIO16 |
| Motor EN | ESP-32 | GPIO27 |
| Motor Step | ESP-32 | GPIO14 |
| Motor Direction | ESP-32 | GPIO12 |
| Solenoid 1 | ESP-32 | GPIO33 |
| Solenoid 2 | ESP-32 | GPIO32 |

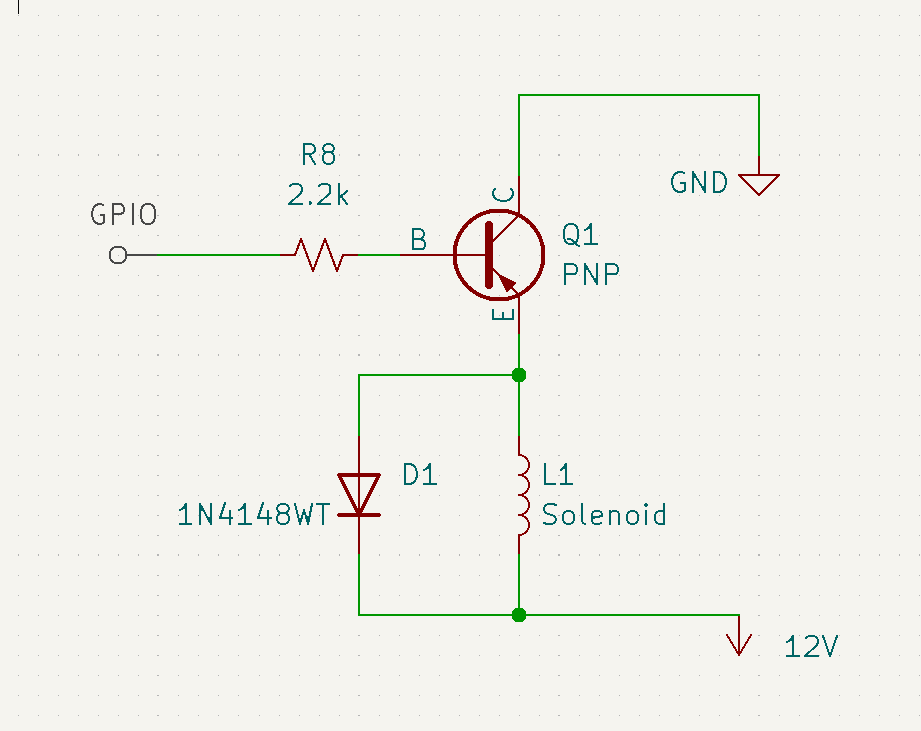
### 3.1.3 Solenoid Circuit Design

Figure 5 Initial Solenoid Circuit



The solenoid circuit in the Automatic Connect Four system controls token release, driven by the ESP32. Figure 5 shows the initial solenoid circuit, using one NPN transistor (2N3906), one MOSFET (IRF540), resistors (2.2kΩ, 470Ω \*2), and a 5V supply, with the solenoid powered by 12V. This design faced challenges due to high current drawing, causing the transistors to overheat and inconsistent triggering from voltage drops (Baira & Koirala, 2025). Figure 6 presents the final solenoid circuit, simplified to one PNP transistor, a 2.2kΩ resistor, and a flyback diode (1N4148W), directly connected to ESP32 GPIO pins 32 and 33 (tmc2209.py, Baira & Koirala, 2025). This reduced component counts improved reliability, lowered heat, and ensured consistent token release triggered via the solenoid function (main.py) (Baira & Koirala, 2025).

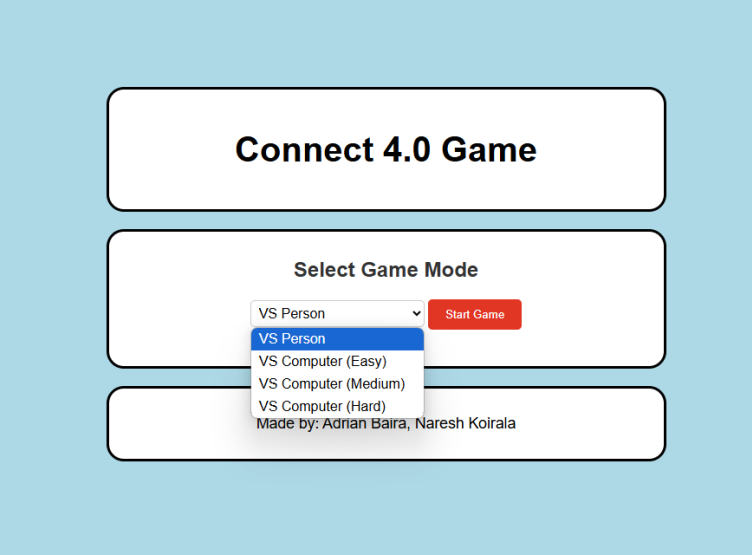
Figure 6 Final Solenoid Circuit



## 3.2 Software Development

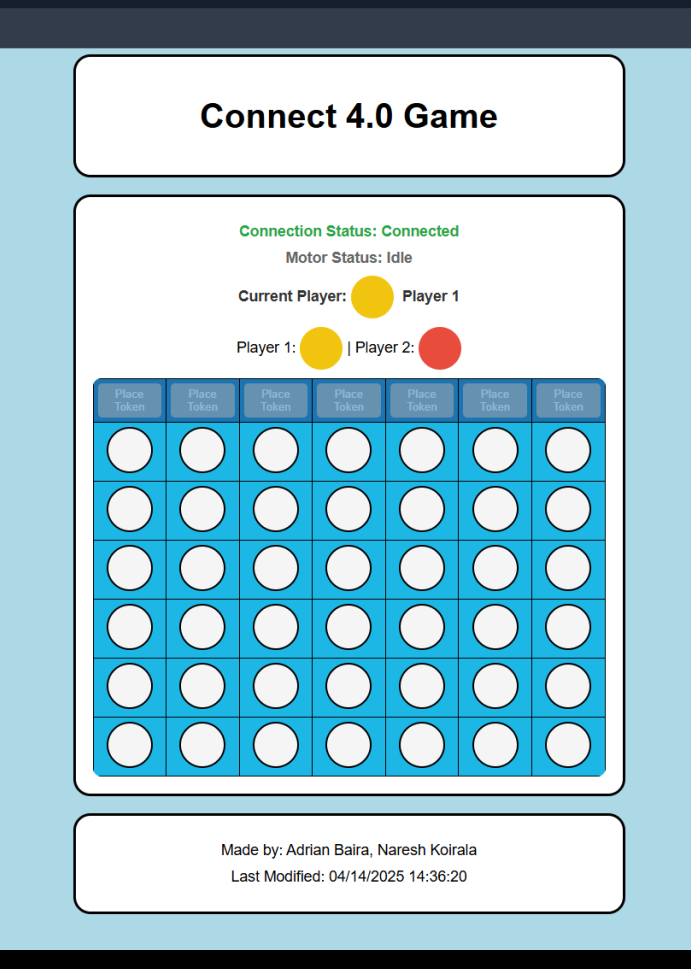
### 3.2.1 Web Interface Development

Figure 7 Website Home Page



This project uses the ESP32 microcontroller to host a website, allowing Player 2 to play Connect Four on a custom gameboard. The website, which is served via website.py, displays a 7x6 Connect Four board with clickable column buttons for easy interaction (script.js). It updates in real-time, visually reflecting token drops and game progress using the play function, which syncs with the physical board via MQTT (main.py). The interface also supports game resets (resetGame) and difficulty selecting computer modes, enhancing user engagement across solo and two-player modes (Baira & Koirala, 2025).

Figure 8 During the Game

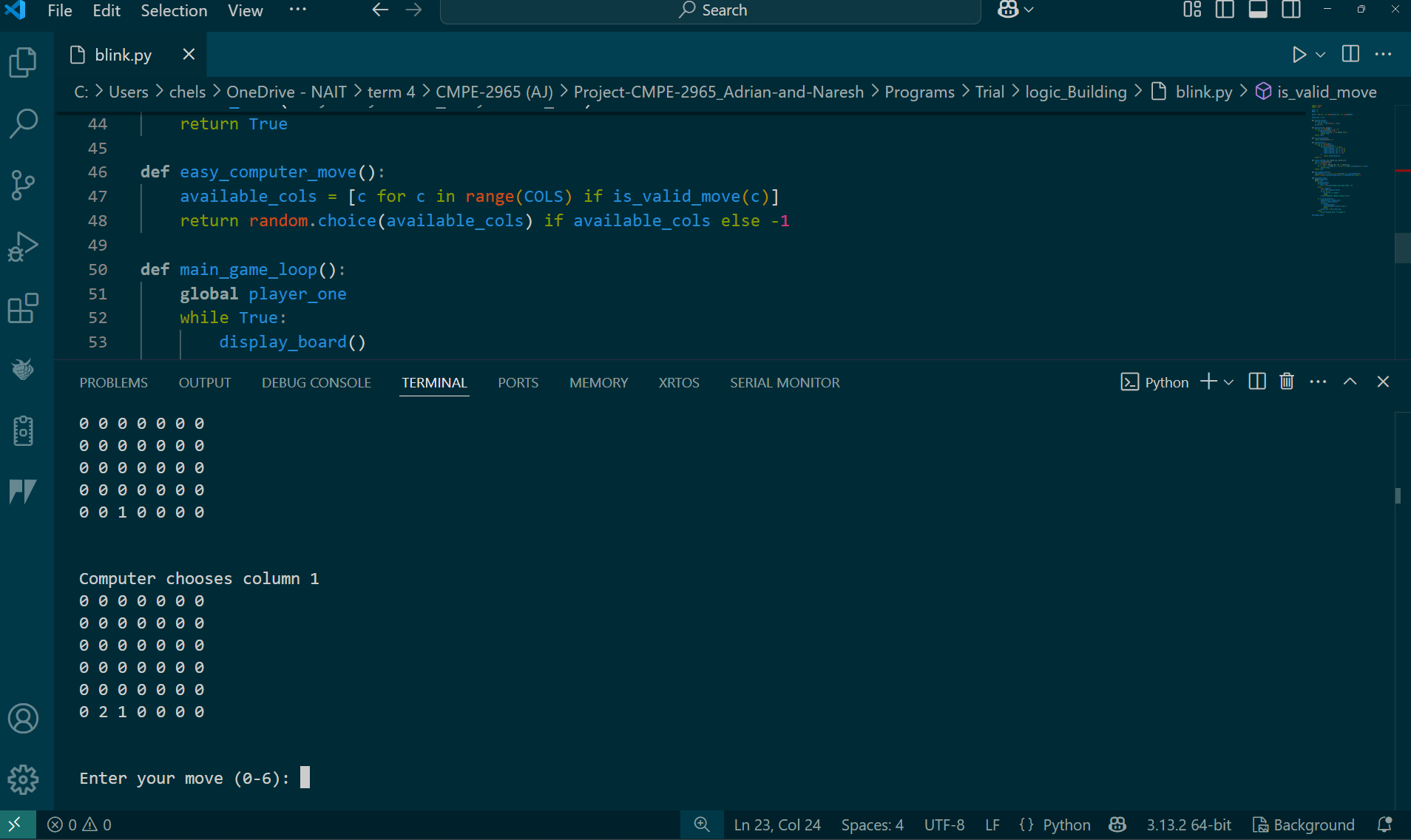


### 3.2.2 Client-side Development

The website enabled users to interact with the Connect Four game through a user-friendly web interface. This project used HTML to structure the website (W3School, n.d), while CSS was used to style its elements and provide a clean and visually appealing layout (W3Schools, n.d). JavaScript handled game logic and user inputs, allowed users to place game pieces and select different game modes, and updated the interface to mirror the ongoing gameplay (W3Schools, n.d). MQTT protocol simplified real-time communication between the website and the physical gameboard (OASIS, 2023). This ensured that the game state remained synchronized and that users could play seamlessly. JavaScript also handled user inputs, such as placing game pieces, and updated the interface to reflect the ongoing gameplay (W3Schools, n.d).

### 3.2.3 Game Logic Development

Figure 9 Initial Game Logic Code



The game logic for the Automatic Connect Four system manages the core gameplay mechanics, including board setup, move validation, and win detection. Implemented primarily on the Pico W, the logic uses a 7x6 board matrix (game\_logic.py), where setup\_board initializes the board, make\_move places tokens by column, and check\_winner detects four-in-a-row across horizontal, vertical, and diagonal lines. ESP32 complements this by updating the web UI (script.js), tracking moves via play and resetting the game with resetGame after a win. Player turns alternate between 1 (Yellow) and 2 (Red), with Pico W handling Player 1’s sensor inputs and the ESP32 managing Player 2’s web inputs (main.py for both). Computer modes (easy/medium) use random column selection (easy\_computer\_move), with plans for enhanced AI (Baira & Koirala, 2025).

## 3.3 Physical Component

### 3.3.1 Pico W

Figure 10 Pico W

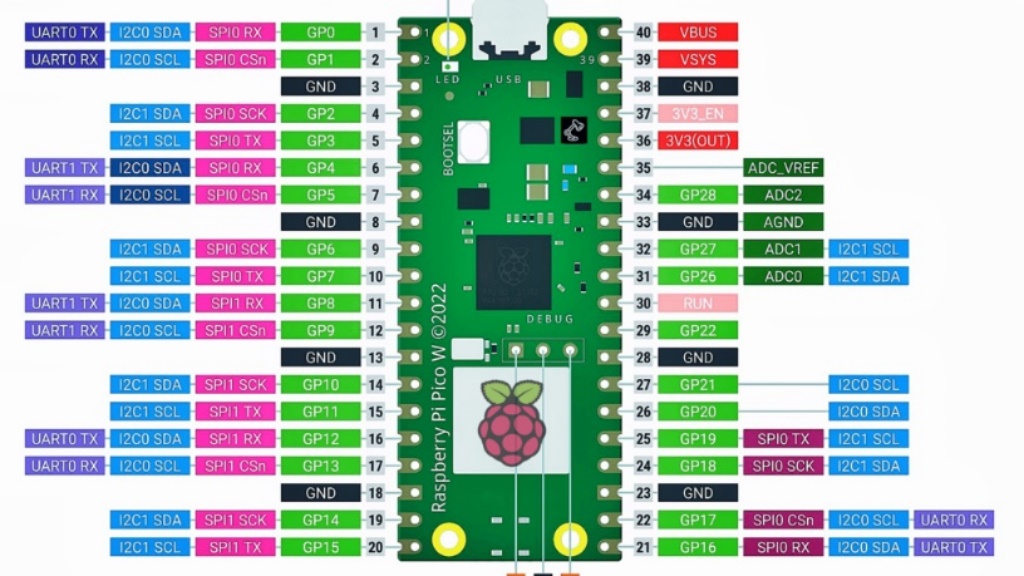


Figure Sourced from rajivcodelab (2024)

The Raspberry Pi Pico W manages sensor inputs and game logic in the Automatic Connect Four system. It uses GPIO pins 16-22 to interface with laser break sensors, detecting token drops for Player 1 (sensor.py). Pico W connects to Wi-Fi via a mobile hotspot (connect\_wifi.py) to communicate with the ESP32 over MQTT, publishing column data (main.py). It also handles game state, tracking moves and winners on a 7x6 board (game\_logic.py). Pico W’s role ensures reliable sensor-based input for hardware play modes (Baira & Koirala, 2025).

### 3.3.2 ESP-32

Figure 11 ESP-32

A diagram of a circuit board

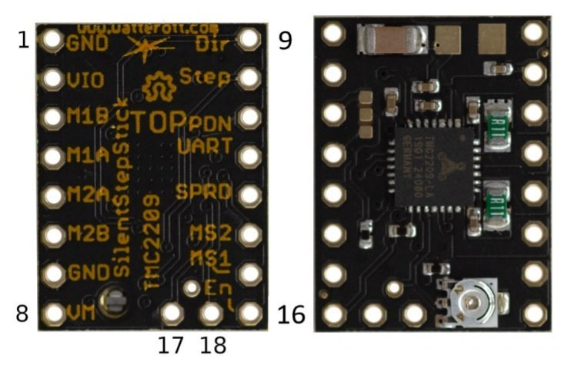
AI-generated content may be incorrect.

Sourced from ESP32 (2024)

The ESP32 controls the token-dropping mechanism and hosts the web interface for Player 2 in the Automatic Connect Four system. It interfaces with the TMC2209 stepper motor driver via pins 14 (STEP), 12 (DIR), and 27 (EN) to position the token dropper and pins 33 and 32 for solenoid control (tmc2209.py). The ESP32 connects to Wi-Fi (connect\_wifi.py) and uses MQTT to receive column data from the Pico W, triggering motor movements (main.py). It also serves the web UI (website.py), enabling player vs. player and computer modes (Baira & Koirala, 2025).

### 3.3.3 Stepper Motor Driver

Figure 12 TMC 2209



MicrocontrollersLab. (2021, March 11)

The TMC2209 stepper motor driver controls the token-dropping mechanism in the Automatic Connect Four system, enabling precise movement across the game board's columns via the ESP32. It is selected for efficiency and low heat, ensuring accurate token placement with a 1 mm precision (Baira & Koirala, 2025). Figure 5 shows the TMC2209’s pin layout, with STEP (pin 10), DIR (pin 9), and EN (pin 16) connected to ESP32 pins 14, 12, and 27, respectively (tmc2209.py). The driver moves 1342 steps per column, as set in move\_val, and controls solenoids (pins 33, 32) to release tokens (main.py). This setup supports reliable operation in both game modes, mitigating overheating issues.

### 3.3.4 SENSOR THRU-BEAM 508MM OPEN COL

Figure 13 5 mm IR break-beam sensor pair for token detection (Adafruit, 2018).



**DigiKey. (n.d.).**

The Automatic Connect Four system uses 5 mm IR break-beam sensors to detect token drops across the board’s columns. Figure 6 shows the emitter and receiver pair, which operate over a 50 cm range with a response time under 2 MS (Adafruit, 2018). Powered at 5V for optimal range, the sensors draw 20 mA, with the receiver outputting via an open-collector transistor, sinking to 100 mA (2168\_Web.pdf). The Pico W interfaces with seven sensors on GPIO pins 16-22, triggering interrupts on beam breaks (sensor.py). This setup, mounted at each column's base, achieved 97% detection accuracy during testing (main.py; Baira & Koirala, 2025).

# 4.0 Implementation

The automatic Connect 4 board was developed by combining hardware, mechanical, and software components, followed by regular testing to verify functionality. This section outlines the assembly process, software syncing, and minor 3D design improvements while managing key challenges during this project development.

## 4.1 Hardware and Mechanical Assembly

The hardware setup stuck to the KiCad schematic (Figure 4). The Raspberry Pi Pico W was wired to seven laser break sensors, connected to GPIO16–GPIO22 (Table 1), each with a 10kΩ pull-up resistor to maintain a stable 5V signal. A 3D-printed frame mounted on the Connect 4 board secured the sensors for accurate token detection. The ESP32 controlled a 12V stepper motor via an A4988 driver, using GPIO14 (step), GPIO12 (direction), and GPIO27 (enable) pins linked to a belt-driven rail system for token positioning. A 12V solenoid, managed by GPIO32 and GPIO33 through a transistor circuit with flyback diodes, released tokens. Power was supplied by a 12V source for the motor and solenoid, while the microcontrollers used USB during the initial setup. Mechanically, a 3D-printed cylinder token holder (Figure 3) with a 45-degree ramp to guide the token into the holder. The rail system moved the token holder across columns. Designing this rail system proved challenging, as initial ideas struggled with token holder stability. Early prototypes displayed inconsistent movement due to belt slack, needing multiple iterations. This was resolved by including a belt tensioner, which gave smooth and reliable positioning across all columns.

## 4.2 Software Integration

The Raspberry Pi Pico W, programmed in MicroPython, monitored sensor inputs on GPIO16–GPIO22, detecting token drops when laser beams were broken. Data was transmitted to the ESP32 using the MQTT protocol over Wi-Fi, facilitated by the MQTT—simple library. The ESP32, coded in MicroPython, handled game logic, motor control, and solenoid activation and hosted a web server. The web interface, built with HTML, CSS, and JavaScript, enabled column selection and real-time updates. Significant trouble arose with MQTT communication and synchronizing the Pico W and ESP32. Wi-Fi was the biggest problem in this project. Both microcontrollers could not connect to the NAIT Wi-Fi, so for this project, we can only work with it on mobile hotspots, which in turn caused multiple packet losses or disconnections when testing, causing significant setbacks in the areas this project was being tested at.

## 4.3 Testing and Refinement

Testing began with a prototype integrating the sensor frame, rail system, and token holder. Sensor accuracy reached 97% after 50 token drops per column, though early tests revealed failures due to dust on laser lenses. After timing adjustments, the stepper motor aligned the token holder within 1mm across columns, but initial rail instability required adding support brackets for precision. The solenoid completed 100 release cycles, with minor timing fixes, although the solenoid tends to get stuck because the spring is weak, affecting its reliability. Frequent web interface tests confirmed that column selections triggered actions in under 5 seconds using MQTT messages. The final design kept the original board's structure, ensuring portability and aesthetics—testing outcomes, documented in section 5.0.

# 

This Automatic Connect Four system successfully automated gameplay and enhanced user engagement, meeting its core objectives. The laser break sensors, linked to Pico W GPIO pins 16-22, achieved 97% accuracy in detecting token drops across all columns, with minor issues from pull-up resistors resolved by adding a 100-ohm resistor(sensor.py; Baira & Koirala, 2025). The ESP32-controlled stepper motor, using the TMC2209 driver, delivered 1 mm precision for token placement over 1342 steps per column (tmc2209.py; Baira & Koirala). Solenoids on GPIO pins 32 and 33 were used for consistent token release, keeping seamless gameplay (main.py). The web interface, hosted on the ESP32, provided real-time updates with a 5-second response for column pick (script.js). A significant issue was MQTT's packet loss from mobile hotspot use; however, using prioritized messaging allowed the project to have a reduced packet loss and helped minimize delays. The system repurposed the gameboard effectively, offering an experience in solo and two-player modes, with AI difficulty enhancements planned (Baira & Koirala, 2025).

# 6.0 Conclusion

This Automatic Connect Four project successfully transformed the classic game into an automated, interactive experience, achieving its goals of automating gameplay and enhancing user engagement. The system integrated the Pico W and ESP32 to manage sensors, motors, and a website, delivering 97% sensor accuracy and 1 mm motor precision (sensor.py, tmc2209.py). Challenges like MQTT packet loss due to mobile hotspot use were stopped via prioritized messaging, ensuring smooth gameplay (main.py). The web interface provided real-time updates, making the game engaging for solo and two-player modes (script.js). While the project repurposed the gameboard effectively, future improvements include enhancing the AI bot's difficulty levels and exploring stable Wi-Fi solutions for better synchronization (Baira & Koirala, 2025).

# 7.0 References

**Adafruit Industries. (2018 December 12). *C140-A datasheet* [Image].** [**https://cdn-shop.adafruit.com/product-files/324/C140-A+datasheet.jpg**](https://cdn-shop.adafruit.com/product-files/324/C140-A+datasheet.jpg)

**Adafruit Industries. (2021, May 28). *2168 datasheet* [PDF].**

[**https://www.adafruit.com/product/2168**](https://www.adafruit.com/product/2168)

**Andreas Spiess. (2021, February 21). *#372 How to use the two cores of the Pi Pico? And how fast are interrupts?* [Video]. YouTube.** [**https://www.youtube.com/watch?v=9vvobRfFOwk**](https://www.youtube.com/watch?v=9vvobRfFOwk)

**Electronoobs. (2023, June 18). *Stepper drivers A4988 vs TMC silent drivers - sound & signals* [Video]. YouTube.** [**https://www.youtube.com/watch?v=do1X3KTMn20**](https://www.youtube.com/watch?v=do1X3KTMn20)

**Espressif Systems. (n.d.). *ESP32-WROOM-32 datasheet* [PDF].** [**https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32\_datasheet\_en.pdf**](https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf)

**Esp32 Wroom 32d . (2024, January 30). *ESP32-WROOM-32D Arduino IDE board*.** [**https://ontremdkmiwiring.z21.web.core.windows.net/esp32-wroom-32d-arduino-ide-board.html**](https://ontremdkmiwiring.z21.web.core.windows.net/esp32-wroom-32d-arduino-ide-board.html)

**Hacker Twins. (2023, May 23). *Make a belt-driven linear rail operated by stepper motor* [Video]. YouTube.** [**https://www.youtube.com/watch?v=IySOT4QmRbY**](https://www.youtube.com/watch?v=IySOT4QmRbY)

**Lutz. (2022, July 22). *Step-by-step Pi Pico W: WIFI, GPIO control and webserver* [Video]. YouTube.** [**https://www.youtube.com/watch?v=fqsOZ\_FVgfg**](https://www.youtube.com/watch?v=fqsOZ_FVgfg)

**Making Stuff with Chris DeHut. (2022, May 26). *Using PICO interrupts* [Video]. YouTube.** [**https://www.youtube.com/watch?v=9Fio0BlMsVM**](https://www.youtube.com/watch?v=9Fio0BlMsVM)

**MicrocontrollersLab. (2021, March 11). *TMC2209 stepper motor driver module pinout, features and interfacing*.** [**https://microcontrollerslab.com/tmc2209-stepper-motor-driver-module/**](https://microcontrollerslab.com/tmc2209-stepper-motor-driver-module/)

**MicroPython. (2000). *umqtt.simple* [GitHub repository]. GitHub.** [**https://github.com/micropython/micropython-lib/tree/master/micropython/umqtt.simple**](https://github.com/micropython/micropython-lib/tree/master/micropython/umqtt.simple)

**Mozilla Developer Network. (2024). *JavaScript*.** [**https://developer.mozilla.org/en-US/docs/Web/JavaScript**](https://developer.mozilla.org/en-US/docs/Web/JavaScript)

**OASIS. (2019, March 7). *MQTT version 5.0 specification*.** [**https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html**](https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html)

**Product Design Online. (2019, September 7). *Fusion 360 tutorial for absolute beginners (2020)* [Video]. YouTube.** [**https://www.youtube.com/watch?v=qvrHuaHhqHI**](https://www.youtube.com/watch?v=qvrHuaHhqHI)

**Rajiv Code Lab. (2024, June 25). *Raspberry Pi Pico W pinout explained*.** [**https://rajivcodelab.com/raspberry-pi-pico-w-pinout-explained/**](https://rajivcodelab.com/raspberry-pi-pico-w-pinout-explained/)

**Raspberry Pi. (2022). *Raspberry Pi Pico W datasheet*.** [**https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf**](https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf)

**Raspberry Pi Foundation. (2024, November 25). *Connecting to the internet with Raspberry Pi Pico W* [PDF].** [**https://datasheets.raspberrypi.com/picow/connecting-to-the-internet-with-pico-w.pdf**](https://datasheets.raspberrypi.com/picow/connecting-to-the-internet-with-pico-w.pdf)

**Raspberry Pi Foundation. (2022, June 15). *Get started with Raspberry Pi Pico W* [Webpage].** [**https://projects.raspberrypi.org/en/projects/get-started-pico-w/2**](https://projects.raspberrypi.org/en/projects/get-started-pico-w/2)

**Rayming Technology. (2024, April 25). *Stepper motor vs. DC motor: How are they different?* LinkedIn.** [**https://www.linkedin.com/pulse/stepper-motor-vs-dc-how-different-rayming-techonloy-onrbc**](https://www.linkedin.com/pulse/stepper-motor-vs-dc-how-different-rayming-techonloy-onrbc)

**W3Schools. (n.d). *CSS tutorial*.**

[**https://www.w3schools.com/css/**](https://www.w3schools.com/css/)

**W3Schools. (n.d). *Html tutorial*.**

[**https://www.w3schools.com/Html/**](https://www.w3schools.com/Html/)

**W3Schools. (n.d). *Java tutorial*.**

[**https://www.w3schools.com/js/default.asp**](https://www.w3schools.com/js/default.asp)