Review the assignment due date

final-project-skeleton

Team Number: Team25

Team Name: GANHEMT

Team Member Name	Email Address
Tanxuan Li	jeffli88@seas.upenn.edu
Xiao Wang	wang96@seas.upenn.edu
Zhenyao Liu	liu59@seas.upenn.edu

GitHub Repository URL: https://github.com/upenn-embedded/final-project-f25-f25-final_project_t25.git

GitHub Pages Website URL: [for final submission]*

Final Project Proposal

1. Abstract

Our project aims to design and implement an interactive desktop pet robot that can engage in real-time communication with users through voice, visual, and motion responses. The system is built on an ATmega328PB microcontroller, integrating an offline voice recognition module, a temperature and humidity sensor, a TFT LCD display, a DFR0299 voice playback module, micro servos, and a Wi-Fi module.

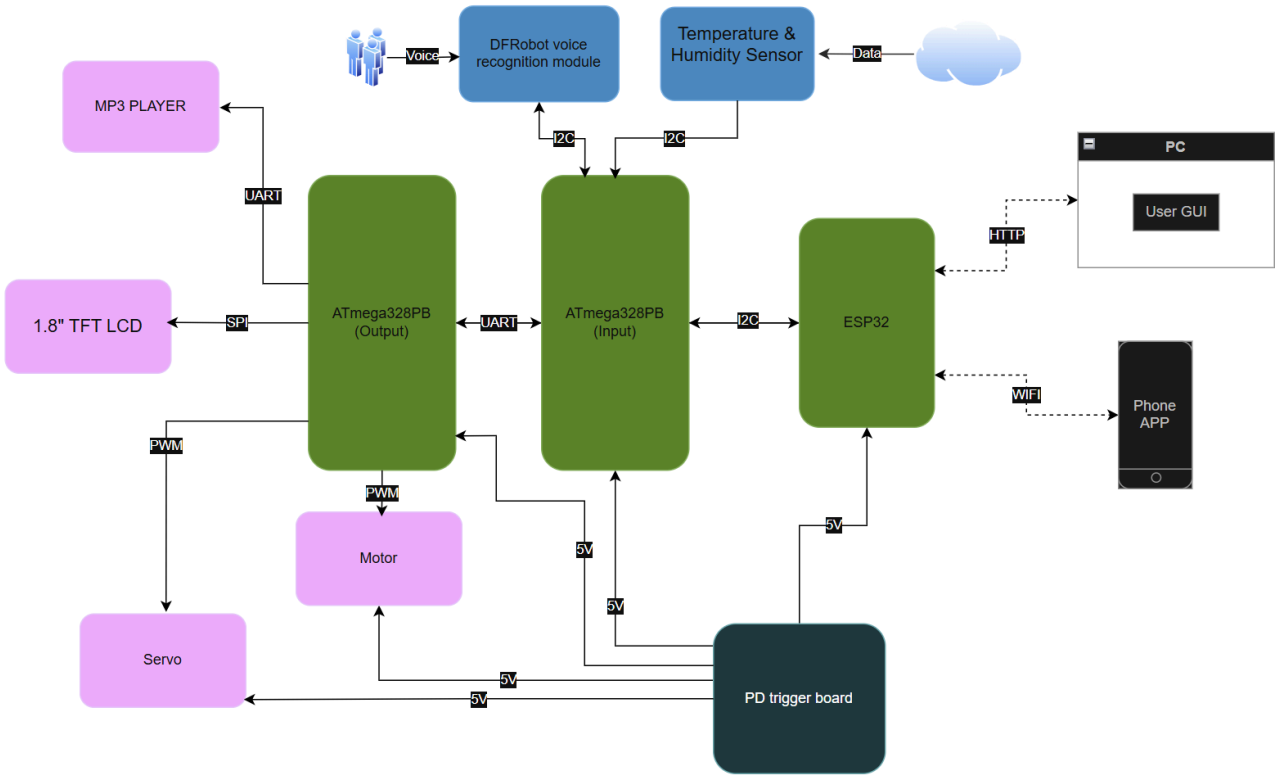
The robot can recognize voice commands, respond with synthesized or pre-recorded speech, and display dynamic facial expressions and environmental information such as temperature and humidity on the LCD. Additionally, the Wi-Fi module allows users to control the robot through a mobile application, providing an alternative way to interact and customize behaviors.

2. Motivation

In modern life, people increasingly spend long hours studying or working alone at their desks, often leading to stress and a lack of social interaction. Our project aims to design a smart desktop companion that provides emotional engagement and environmental awareness through human–machine interaction. By combining voice recognition, facial expression display, and motion response, the robot creates a friendly and interactive experience that brings liveliness to a personal workspace.

3. System Block Diagram

System Block Diagram:



High-level design (per figure): a two-MCU architecture separates *sensing/comms* from *actuation/UI* . The **Input MCU (ATmega328PB)** aggregates sensors and bridges to Wi-Fi; the **Output MCU (ATmega328PB)** drives motors, servos, display, and audio. An **ESP32** provides Wi-Fi/HTTP links to a **PC GUI** and **phone app** . Power comes from a USB-C PD power bank/brick via a PD trigger board.

Critical Components & Roles

- **ATmega328PB (Input)**
  - Collects **DFRobot offline voice** (I^2C) and **Temp/Humidity sensor** (I^2C).
  - Exchanges commands/status with ESP32 (I^2C).
  - Sends parsed commands to Output MCU (**UART**).

- **ATmega328PB (Output)**
  - Drives **1.8″ TFT LCD** (**SPI**) for eyes/status.
  - Controls **motors** (**PWM**) and **servos** (**PWM**).
  - Plays feedback audio via **DFPlayer MP3** (**UART**).
- **ESP32**
  - **Wi-Fi** + **HTTP** to PC GUI / phone app.
  - Local bridge between network and Input MCU (I^2C).
- **DFRobot Voice Recognition Module**
  - Local, offline command recognition (I^2C input → events to Input MCU).
- **Temp & Humidity Sensor**
  - Environmental telemetry (I^2C to Input MCU).
- **DFPlayer Mini + Speaker**
  - Audio prompts and reactions (UART from Output MCU).
- **Actuators**
  - **4× DC motors** (PWM from Output MCU via motor drivers).
  - **2× micro servos** (PWM from Output MCU).

Communication Map

- **I^2C**
  - Input MCU ↔ Voice module
  - Input MCU ↔ Temp/Humidity sensor
  - Input MCU ↔ ESP32 (command/status bridge)
- **UART**
  - Input MCU ↔ Output MCU (bidirectional command channel)
  - Output MCU → DFPlayer Mini (audio control)
- **SPI**
  - Output MCU → 1.8″ TFT LCD
- **PWM**
  - Output MCU → 4× Motor driver PWM inputs
  - Output MCU → 2× Servo signal lines
- **Wi-Fi / HTTP**
  - ESP32 ↔ PC GUI / Phone App (remote panel, telemetry)

Power Regulation

- **Source:** USB-C Power Delivery (PD) power bank; PD Trigger Board, negotiates a fixed PD voltage (5v) and exposes it as a stable DC output.

- **Decoupling:** ⩾470–1000 µF on motor rail; ⩾100 µF near ESP32; 0.1 µF at each IC.
- **Grounding:** common ground star-point; keep ESP32 antenna and mic away from motor wiring.

Data/Control Flow

1. User speaks → Voice module (I^2C) → Input MCU parses → sends CMD via UART to Output MCU.
2. Remote panel (PC/phone) → HTTP over Wi-Fi → ESP32 → I^2C to Input MCU → UART to Output MCU.
3. Output MCU executes: PWM motors/servos, updates TFT (SPI), triggers DFPlayer (UART).
4. Status/telemetry (sensor values, heartbeats) flow back to ESP32 → HTTP → PC/phone UI.

4. Design Sketches

- **Overall Appearance:**

  The system is designed as a **desktop companion robot (“desk pet”)** . It reacts to human voice and remote commands, displaying animated “eye” expressions on its TFT screen and responding through motion and sound.

- **Functional Layout (see figure above):**

  - **ATmega328PB (Input MCU)** collects data from the **voice recognition sensor** and **temperature/humidity sensor** , then communicates with the **ESP32** via I^2C.
  - **ESP32** handles Wi-Fi communication with the **PC User GUI** and **Phone App** via HTTP/Wi-Fi.
  - **ATmega328PB (Output MCU)** drives the **TFT display** , **DFPlayer MP3 module** , **motors** , and **servos** using SPI, UART, and PWM interfaces.
  - **Power** is provided by a USB-C PD source through a PD trigger board that outputs a fixed DC 5V voltage.

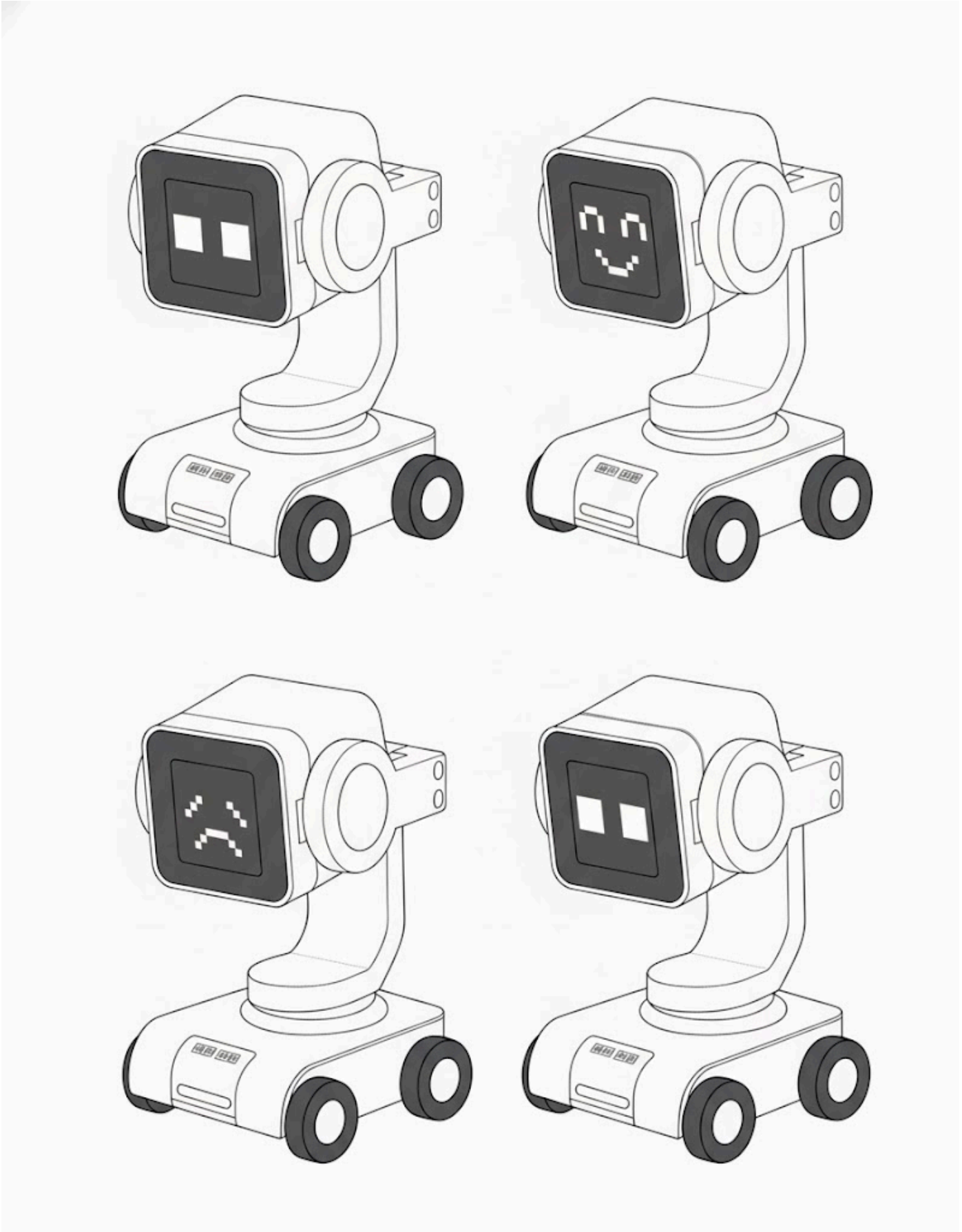
- **Critical Design Features:**

    - Compact and **cute robotic form** that expresses emotions through the screen and sound.
    - **Offline speech recognition** for direct command response without internet.
    - **Wi-Fi remote control** via PC and phone dashboard for movement and reaction commands.
    - Integrated **multimodal feedback** : voice, motion, and display.

- **Manufacturing & Assembly:**

    - **3D printing** will be used to create the outer shell and mounting brackets.
    - The structure will be assembled on an **acrylic baseplate** for stability.
    - **No advanced machining** required—hand tools and a 3D printer are sufficient.

Sketch for this project:



5. Software Requirements Specification (SRS)

5.1 Definitions, Abbreviations

- **MCU**: ATmega328PB microcontroller
- **DFR0299**: MP3 module, controlled via UART serial commands
- **LCD (ST7735)**: 1.8" 128×160 SPI TFT color display controller, SPI protocol
- **AHT20**: Adafruit AHT20 Temperature & Humidity Sensor, I^2C interface
- **Micro Servo**: TowerPro SG92R, PWM control
- **ASR**: Automatic Speech Recognition
- **NDJSON**: Newline-Delimited JSON stream protocol for HTTP communication
- **CMD**: Control Command (MOVE, STOP, TURN, SERVO, etc.)
- **UI**: Remote control user interface (PC web or mobile web/app)

5.2 Functionality

ID	Description
SRS-01	The AHT20 temperature reading shall retain 1 °C resolution, and the humidity display shall retain 1 % RH resolution.
SRS-02	LCD partial area refresh shall complete within ⩽ 50 ms.
SRS-03	After a normal playback trigger, the DFR0299 shall transmit a complete command frame to the MCU and receive acknowledgement (or busy signal) within ⩽ 500 ms.
SRS-04	Micro Servo actived by 50Hz pulse, then read the duty cycle and respond to the a certain angle basic on the duty cycle value.
SRS-05	The offline speech recognition module shall recognize at least 10 predefined voice commands (e.g., forward, back, left, right, stop, hello) with an accuracy ⩾ 85 % under indoor noise ⩽ 65 dB. The end-to-end latency (from end of speech to command reception by MCU) shall be ⩽ 600 ms.
SRS-06	Communication between MCU and PC (via ESP32/HTTP bridge) shall implement a sequence-acknowledgment mechanism.
SRS-07	The remote control panel (desktop & mobile) shall provide MOVE/STOP/TURN/SERVO/EMOTE/SAY controls. The visible feedback latency (UI action → MCU acknowledgment) shall be ⩽ 200 ms (95th percentile).

6. Hardware Requirements Specification (HRS)

6.1 Definitions, Abbreviations

- **VCC / 5 V / 3V3**: Power rails
- **MCU**: ATmega328PB microcontroller
- **DFR0299**: MP3 module, controlled via UART serial commands
- **LCD (ST7735)**: 1.8" 128×160 SPI TFT color display controller, SPI protocol
- **AHT20**: Adafruit AHT20 Temperature & Humidity Sensor, I^2C interface
- **Micro Servo**: TowerPro SG92R, PWM control
- 3D **printing**: To print the case and connections
- **ASR**: Automatic Speech Recognition

6.2 Functionality

ID	Description
HRS-01	The AHT20 shall be powered by 3.3 V, with power ripple ⩽ 50 mV pp. The physical connection length should be ⩽ 20 cm and routed away from high dv/dt signals.
HRS-02	The LCD shall display temperature and humidity readings, five facial expressions, and text responses to voice commands.
HRS-03	Upon a normal speech trigger, the DFR0299 shall produce the first audible syllable within ⩽ 1.5 s, and shall be capable of playing ten distinct response phrases.
HRS-04	The micro servo requires a continuous 50 Hz PWM signal (corresponding to a 20 ms period). Within each cycle, the servo angle is precisely determined by the width of the high-level pulse, typically ranging from 1 ms to 2 ms.
HRS-05	The motor controlled by the PWM signal that on and off the MOSFET.
HRS-06	Module–MCU link (I^2C 400 kHz or UART 115200) shall run 10 min error-free (CRC/frame = 0).
HRS-07	The power supply should be able to support MCU and peripheral. Max current>=2A.

7. Bill of Materials (BOM)

ESE5190 F25 Final Project BOM.xlsx - Google Sheets

Core Components

- **ATmega328PB Xplained Mini ×2** – Main MCUs; one handles inputs (sensors, voice, Wi-Fi), the other handles outputs (motors, servos, display).
- **FeatherS2 (ESP32-S2)** – Wi-Fi bridge for HTTP communication with PC/mobile app.

Input & Communication

- **Gravity: Offline Voice Recognition Sensor (DFRobot)** – Local speech command recognition (I^2C).
- **Temperature & Humidity Sensor (e.g., BME280/SHT31)** – Environment telemetry for UI display.

Output & Feedback

- **1.8” 128×160 TFT LCD (ST7735)** – Display robot “eye” and status (SPI).
- **DFPlayer Mini MP3 Module + 3 W Speaker** – Audio output for responses and sound effects.

Actuation

- **Micro Servo (TowerPro SG92R) ×2** – Control eye/head movement (via PCA9685).
- **DC Gear Motor ×4 (TT/N20 type)** – Drive 4 wheels for motion control.
- **TB6612FNG Motor Driver ×2** – Dual-channel drivers, one per motor pair.
- **PCA9685 16-ch PWM Module** – I^2C PWM expander for servos/LEDs.
- **65 mm Multi-Hub Wheels ×4** – Wheels for TT/N20 motors.

Power & Regulation

- **USB-C PD Power Bank** – Primary source (⩾20 W).
- **USB-C PD Trigger Board** – Negotiates and outputs fixed 5 V.
- **Bulk Capacitors (470–1000 µF)** – Filter motor surges and Wi-Fi peaks.
- **Level Shifter (BSS138)** – I^2C 5 V↔3.3 V conversion if needed.
- **Emergency Stop Button** – Safety cutoff for PWM outputs.

8. Final Demo Goals

- **Demonstration Setup**
  - The robot will be placed on a tabletop or smooth indoor floor.
  - Powered by a USB-C PD source via a PD trigger board.
  - Controlled through a **PC web dashboard** and **mobile app interface** over Wi-Fi.
- **Core Demonstrations**
  - **Speech Interaction:**
    - Say voice commands such as *"forward,"* *"stop,"* *"turn left,"* or *"hello"* to trigger immediate robot actions.
    - Display visual feedback (eye animation on TFT screen) and audio confirmation via DFPlayer.
  - **Wi-Fi Communication:**
    - ESP32 bridges commands between the robot and PC/mobile remote panel.
    - Real-time control and status feedback demonstrated through a browser UI.
  - **Remote Control Panel:**
    - Web interface buttons to control motion, servos, and expressions.
    - Command latency and heartbeat reliability shown live (RTT $\leqslant$ 80 ms).
- **Physical Constraints**
  - Indoor space ($\approx$ 2 m $\times$ 2 m) required for safe movement.
  - Flat surface; lightweight chassis; wired USB debugging available.
  - No external network required—ESP32 operates as local access point or on lab Wi-Fi.
- **Evaluation Metrics**
  - Successful local and online speech recognition ($\geqslant$ 85 % accuracy).
  - Stable motion control via remote panel (no packet loss > 1 %).
  - Consistent power operation—no brownouts during Wi-Fi transmission.

9. Sprint Planning

Milestone	Functionality Achieved	Distribution of Work
Sprint #1	Voice Recognition, Speaker, Sensor Reading	It can perform speech recognition, perceive the surrounding environment in multiple aspects, carry out accurate measurements, and enable the speaker to play preset sounds.
Sprint #2	LCD and fixed the bugs last week. First draft of the 3D case. WIFI	The screen can display the corresponding preset expressions, estimated at around 5 to 10 types. Make a prototype of the case based on the components. Able to communicate with the computer through the Wi-Fi module
MVP Demo	Able to move forward and backwards, nod and shake. Second draft of the case.	Can move based on the input instructions.
Final Demo	Assemble and print the 3D case	Assemble everything.

Sprint Review #1

Plan for This Week

1. **ESP32 Wireless Communication:**

   Establish wireless communication between the ESP32 module and the PC through Wi-Fi using HTTP requests. The ESP32 was expected to retrieve information such as current time and weather data from a local server running on the PC, while the PC could monitor the controller’ s state.

2. **LCD Display Exploration:**

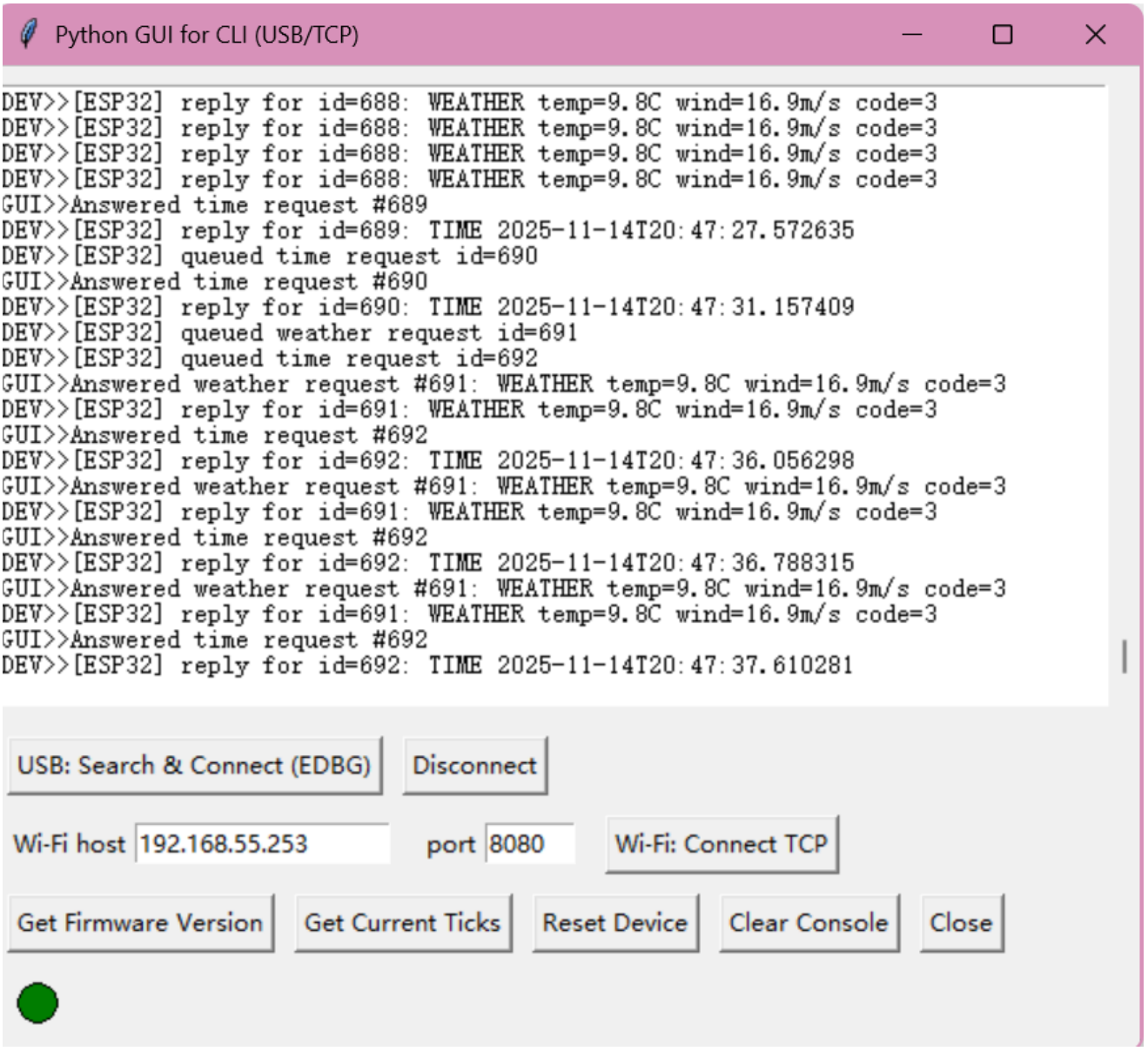
Gain a basic understanding of how the ST7735 LCD works on the ATmega328PB. Learn to draw basic shapes, test simple display functions, and implement an initial “facial expression” demo using primitive graphics.

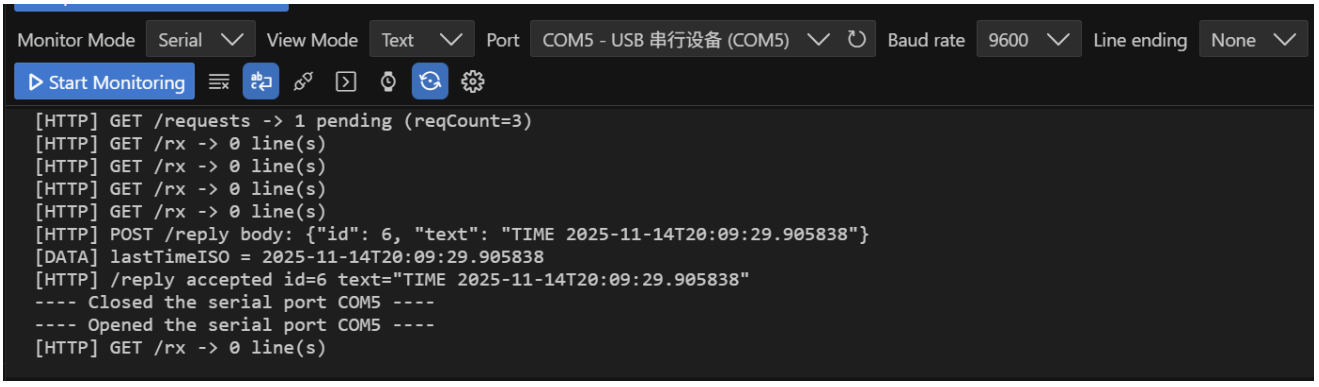
3. **Motion Control Foundation:**

   Begin developing a prototype for the robot’s base and motion control system, focusing on PWM signal generation and servo angle control.

Things Done and Proof

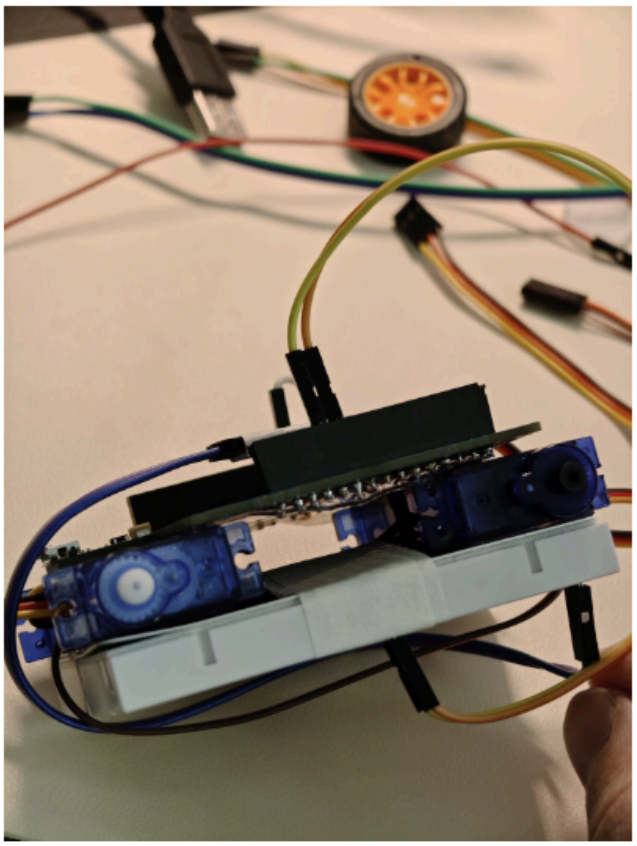
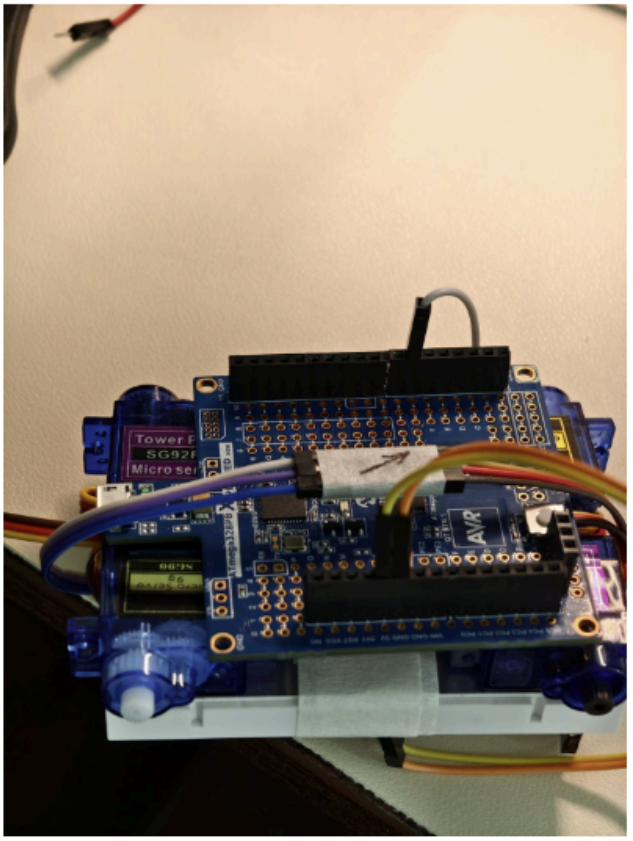
1. **ESP32 Wireless Communication:**
   - Successfully implemented HTTP-based communication between ESP32 and PC over Wi-Fi.
   - Developed a Python server on the PC that responds to ESP32 requests and provides real-time data (time, weather).
   - Verified bidirectional communication:
     - ESP32 sends GET requests to fetch information.
     - PC logs and displays controller state and received requests.
   - Tested multiple times under stable Wi-Fi connection.
   - Proof: Serial logs showing successful HTTP responses; server terminal displaying request traces.





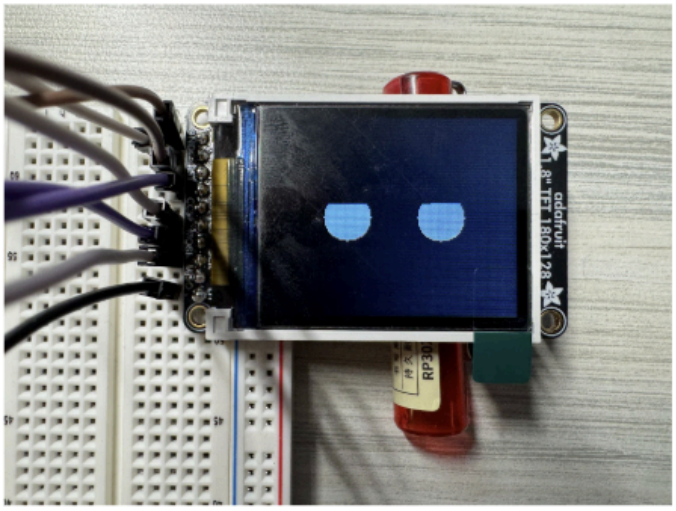
2. **Motion Control System:**

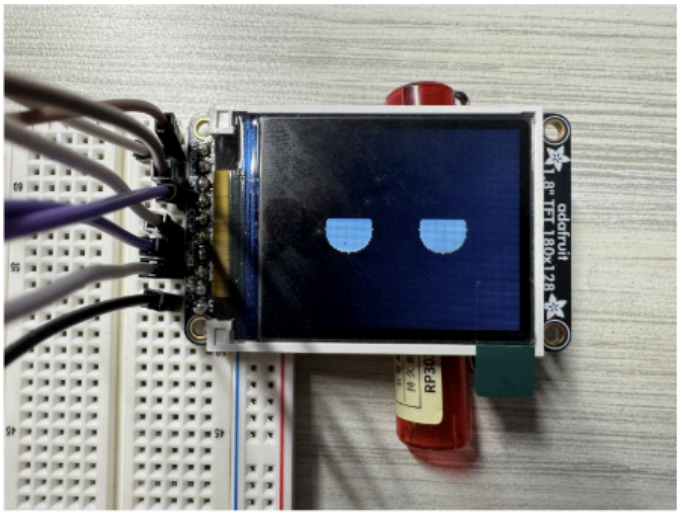
   - **Low-Level Pulse Control:** Implemented functions to adjust PWM pulse width (OCR1A/B), including range validation (2000–4000 counts).
   - **Stable PWM Generation:** Achieved 50 Hz frequency (20 ms period) for standard servo control.
   - **Angle Abstraction Layer:** Developed functions mapping 0–180° input to corresponding pulse widths using linear interpolation.

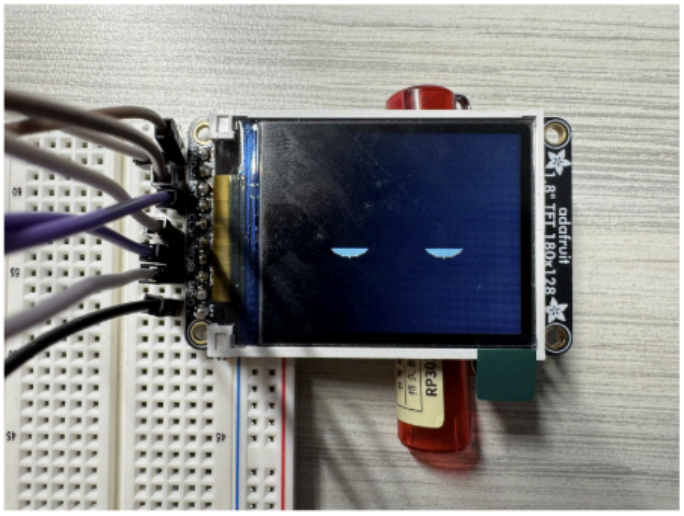



3. **LCD Display & Animation:**

   - Created a **blinking-eye animation** , with eyes closing and reopening using efficient region refreshing.
   - Added synchronized eyebrow movement for enhanced expressiveness.
   - Verified smooth frame transitions and minimal flicker during updates.







Difficulties

1. **ESP32 Communication:**
    - Encountered missing dependencies (`requests`, `ArduinoJson`) on both PC and ESP32.
    - Faced challenges in JSON parsing and Wi-Fi reconnection stability.
    - Resolved through library installation and implementing error-handling/retry mechanisms.
2. **Motion Control Development:**
    - Designing a multi-layer abstraction (Hardware Init → Low-Level Pulse → Mid-Level Angle) required careful modular planning.
    - Developed and tested control logic **without physical hardware** , making validation challenging.
3. **Display Module:**
    - Managing drawing efficiency and synchronization for smooth animation required optimization of display refresh cycles.

Things Not Done

1. **ESP32/Blynk Integration:**
    - Remote control via Blynk not yet implemented.
    - Voice recognition module (VC-02 or DFRobot Offline Speech) setup pending.
    - Full system integration with dual ATmega32 boards not yet performed.
2. **LCD & Sensor Features:**
    - AHT20 temperature/humidity sensor not yet integrated.
    - No real-time data display or UI text elements on LCD.
    - Only the blinking expression implemented; advanced facial expressions (happy, sleepy, surprised) pending.
3. **Motion System & Hardware:**
    - High-level movement logic (forward, backward, turn, stop) functions not yet implemented.
    - No physical case designed for the robot base.

Plan for Next Week

1. **ESP32 Development:**
    - Integrate the Blynk platform for smartphone-based remote control.
    - Begin developing offline voice recognition module.
    - Test full communication flow between ESP32 and dual ATmega32 boards.
2. **LCD & Sensor Expansion:**
    - Read AHT20 temperature/humidity values and display them on the LCD.
    - Design and render text/icons showing environmental data.
    - Implement three additional facial expressions (happy, sleepy, surprised).
3. **Motion & Mechanical Design:**
    - Replace servo motors with DC motors for larger motion range.
    - Design and prototype the robot’ s base casing for mechanical stability.

Sprint Review #2

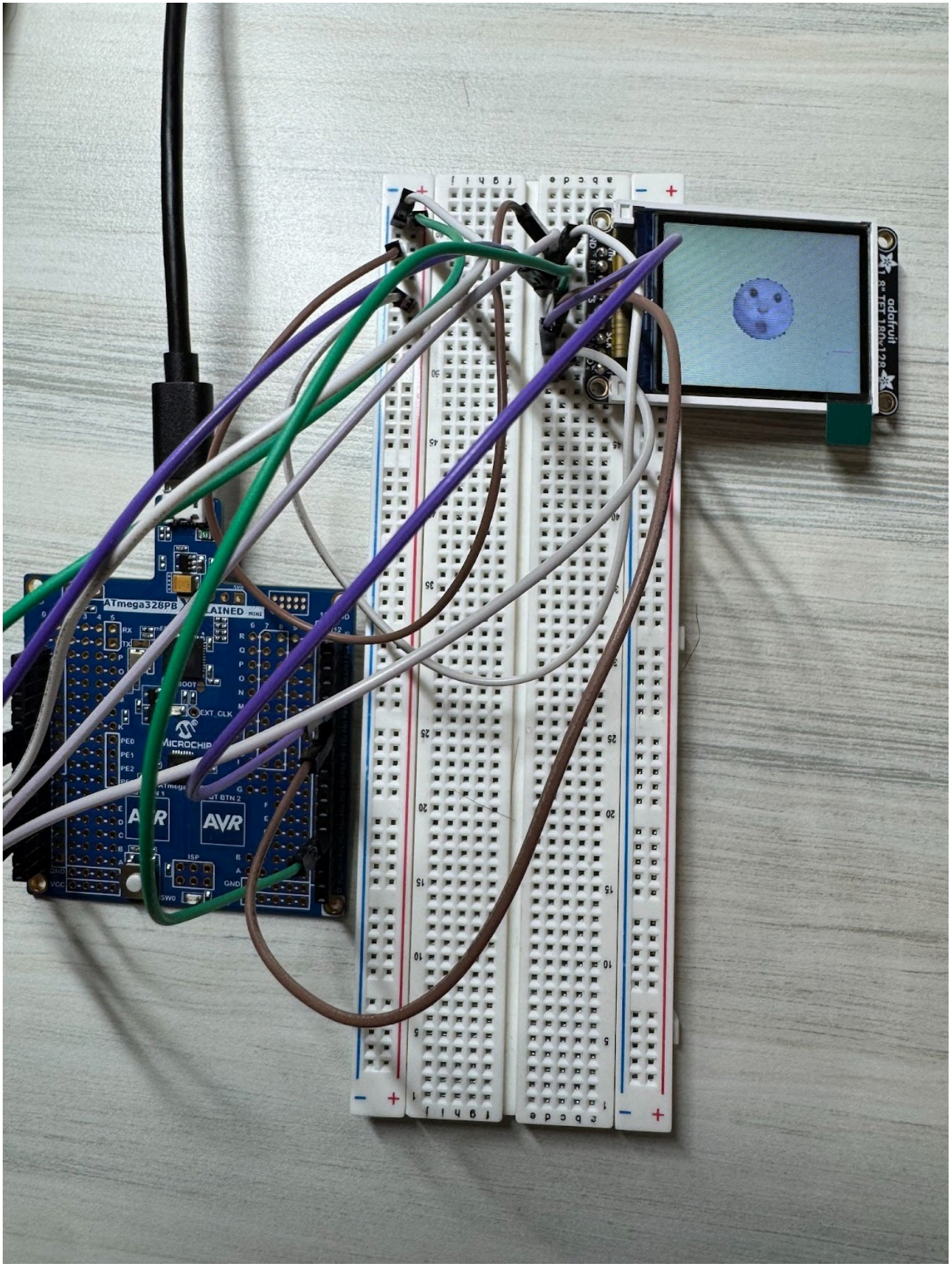
Plan for this week

1. Complete LCD expressions display.
2. Complete the wheel mounting and base control
3. 3.Complete wireless control using Blynk

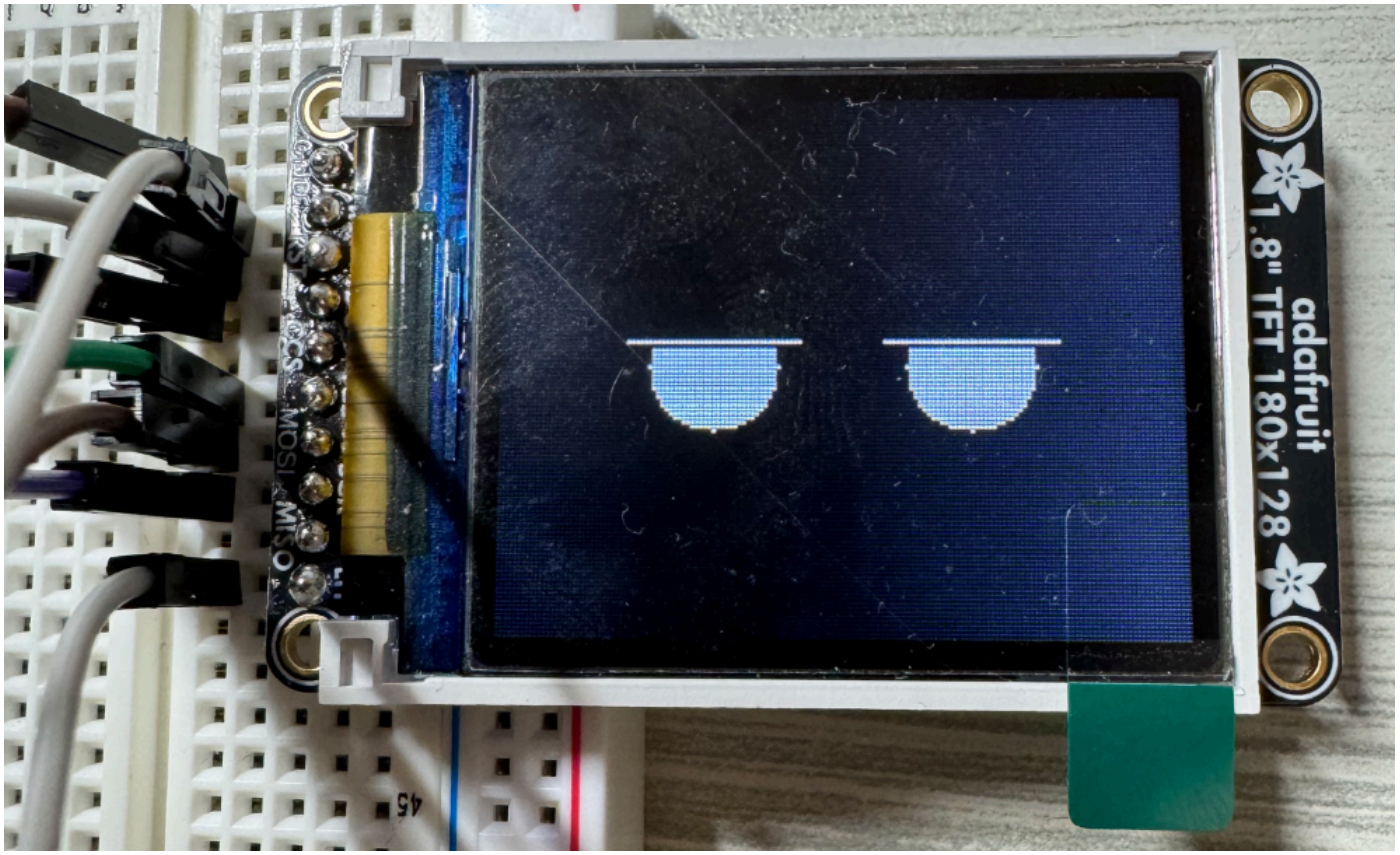
Things done and proof:

1. Focus on LCD part

- Completed multiple LCD expression outputs.
- Refactored the LCD code into a state-machine architecture to avoid blocking and enable smooth multitasking.
- Developed a Python script to convert images into pixel data and successfully displayed them on the LCD.

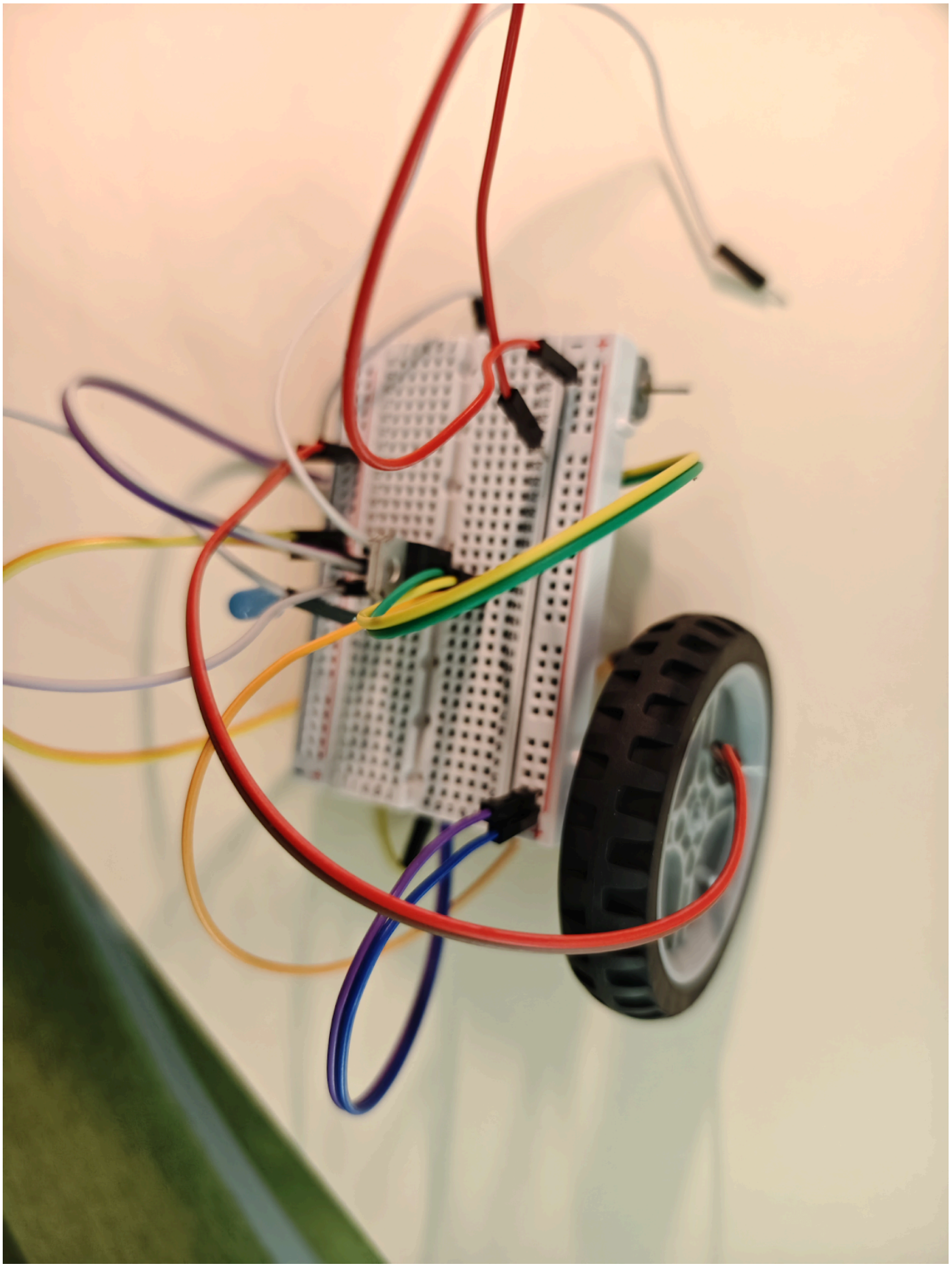


- Implemented the eye-blinking animation with smooth visual performance.



- Resolved LCD hardware wiring issues by rewiring the pins according to the team’ s updated design and verifying successful operation.

2.



- Successfully anchored the wheels to the motor, ensuring reliable mechanical coupling.
- Developed and implemented the control program, enabling precise actuation of movement commands.

- Replaced the servo with a motor to achieve a wider range of motion.
- Verified that the system can now move forward, turn left and right, and adjust wheel speed effectively.

3. Successfully control the motor by Blynk. Use phone to wirelessly control the robot moving.

Difficulty:

1. LCD

- The LCD brightness pin could not be controlled using PWM outside port B, so a normal GPIO pin was used instead.
- LCD drawing operations originally blocked other tasks; solved by implementing a non-blocking state machine that updates pixels line-by-line.

2. Motor:

- Mechanical alignment between the motor and wheel required careful adjustment to avoid slippage.
- Transitioning from servo to motor introduced new challenges in control tuning and stability.
- Ensuring smooth speed regulation demanded iterative testing and parameter calibration.

3. ESP32

- Integration of wireless communication with PC and Blynk.
- Serial communication between ESP32 and ATmega32 and control the motion part for moving.

things not done:

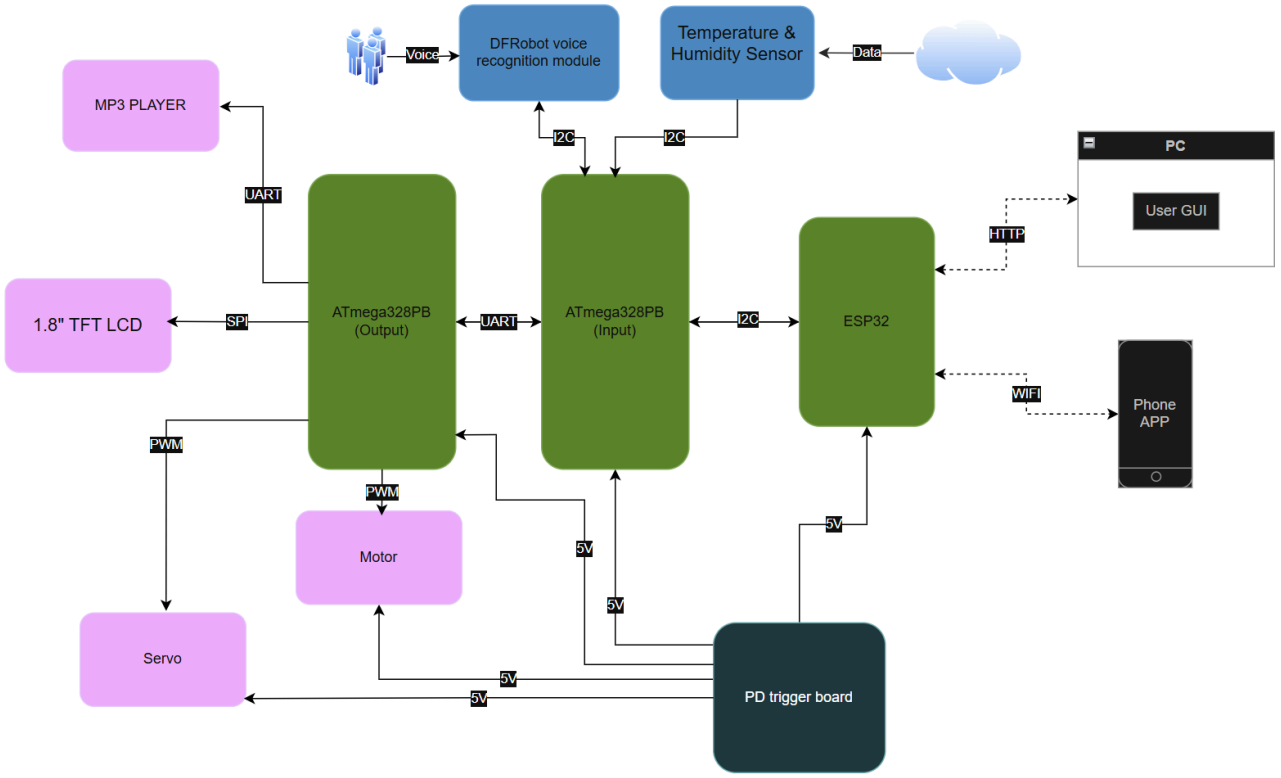
1. Integration of temperature/humidity sensor data and the speaker functionality (blocked due to component issues).
2. Case not added
3. The servo for the head was not added
4. Voice recognition module has not been developed

Plan for next week

1. Implement switching between different expressions, display sensor data on the LCD, and enable the speaker for specific sound playback.
2. Use a plank to build a case for the base
3. Build servo logic for the head.
4. Develop voice recognition module

MVP Demo

1. Show a system block diagram & explain the hardware implementation.



- For LCD, temperature, speaker parts, our system uses an ATmega328PB as the main controller, connected via SPI to a 1.8" ST7735 TFT LCD, via I2C to an AHT20 temperature-humidity sensor, and via UART to a DFPlayer Mini audio module. In our block diagram, the LCD renders facial expressions, the AHT20 provides live environment data, and the DFPlayer plays fixed voice responses when triggered over UART. All parts are powered from the same 5 V rail.
- The motor and servo motor systems are both controlled by the ATmega328PB microcontroller.For the drive motors, we utilize the CTC mode to generate varying frequency waveforms from the controller's output pins. This allows for the precise control of the speed of the two DC motors on each side, enabling the robot to execute basic motion functions such as forward, backward, and turning (left/right).For servo motor control, we employ a continuous 50Hz

PWM signal. The servo's angle is precisely determined by the high-level pulse width: a pulse of 0.5ms corresponds to 0°, and a pulse of 2.5ms corresponds to 180°, allowing for linear interpolation for intermediate angles. We use interrupts in this part as the timer has been used up. The servos are primarily used to implement nodding and head-shaking functionalities, enhancing the robot's interactive responses.The entire system is powered uniformly by a 5V DC power module.

- ESP32 Wi-Fi Communication Module: The ESP32 functions as the wireless communication bridge between the external control interfaces and the robot’ s internal control system. It connects to the local Wi-Fi network and receives commands from either the PC-based server or the Blynk mobile application. These commands are encoded into lightweight serial frames and forwarded to the ATmega328PB via a UART link operating at 115200 bps. In this configuration, the ESP32 is responsible solely for network connectivity and transparent command forwarding, ensuring a clean separation between wireless communication and motor actuation hardware.
- Speech Recognition Module (I^2C Control Interface): An offline speech-recognition module provides a secondary control path. The module operates over the I^2C bus and has been configured with a trained command set tailored to the robot’ s motion primitives. A full TWI communication driver enables the ATmega328PB to poll the module for new recognition events and retrieve command identifiers. The recognized commands are then mapped into the unified control pipeline. This hardware path enables fully local, low-latency voice control without dependency on wireless infrastructure.
- ATmega328PB Motion-Control Interface: The ATmega328PB serves as the central motion-control unit, receiving commands through both UART (from the ESP32) and I^2C (from the speech module). Commands are interpreted and dispatched to the motor driver via PWM and GPIO outputs. The hardware architecture ensures that both wireless and voice-based inputs converge at the ATmega, which is responsible for generating the final actuation signals that drive the robot’ s movement.

2. Explain your firmware implementation, including application logic and critical drivers you've written.

- For LCD part
    - uses a 1 ms Timer0 interrupt to maintain a global millisecond counter (millis()), which is then used to schedule all tasks in a non-blocking way

inside the main while(1) loop
    - For the eyes and eyebrows, the code defines geometry constants and fast SPI drawing helpers (fast_hline, fast_restore_eye_row) that update only horizontal lines or eye rows instead of redrawing the whole screen,A blink state machine (blink_step) is called every 50 ms
    - The mouth and lower-face “expression area” have four modes: smile, sad, pout, and a text screen. Each expression is drawn with a dedicated function using rectangles, circles, and fast horizontal line operations. A global expr_state tracks the current expression. Every 100 ms, the main loop polls UART non-blocking; characters ‘1’ – ‘4’ request expression changes, and ‘5’ – ‘8’ control DFPlayer tracks. A small “switch lock” with timeout prevents very fast expression toggling; when switching, the mouth region is cleared and the new expression is drawn.

- For sensing part
    - the AHT20 is driven over I^2C. Every 500 ms, the code triggers a measurement, reads 20-bit raw temperature/humidity data, converts it into physical values scaled by 10, and accumulates them. Every 10 seconds, it computes average temperature and humidity and updates latest_t_avg10 and latest_h_avg10 plus a sequence counter avg_seq. When the expression is state 4 (T/H screen), any new average triggers an automatic refresh of the bottom text region, where centered strings like “T=xx.x C” and “H=yy.y %RH” are redrawn on the LCD.

- For Motor part
    - the driver utilizes the 8-bit Timer0 in CTC Mode (Mode 2) to generate drive signals on pins PD5 and PD6. Instead of using software loops to toggle pins, the driver dynamically manipulates the COM0A0 and COM0B0 bits to enable hardware toggling on compare match, producing a stable 50% duty cycle square wave. The frequency (which dictates motor speed/pitch) is controlled by OCR0A, allowing the system to switch between FREQ_HIGH and FREQ_LOW. At the application layer, movement is handled by blocking functions like turn_angle, which uses a calibration constant (MS_PER_DEGREE) to convert target angles into precise delay durations, and move_straight_time for linear distance control.

- For Servo part

    - a custom interrupt-driven software PWM driver is implemented using the 16-bit Timer1. Configured with a prescaler of 8, the timer provides 0.5 µs tick resolution with ICR1 setting the standard 20 ms (50 Hz) servo period. The pulse generation relies on three specific Interrupt Service Routines (ISRs): the TIMER1_CAPT ISR sets both Pan (PD3) and Tilt (PD4) pins HIGH at the start of the cycle, while COMPA and COMPB ISRs independently clear the pins when the counter matches the target pulse width. A helper function, angle_to_ticks, maps 0–180° inputs to the required 1000–5000 timer ticks. To prevent servo jitter during updates, the driver performs atomic operations by briefly disabling interrupts (cli()) when updating the global pulse width variables.

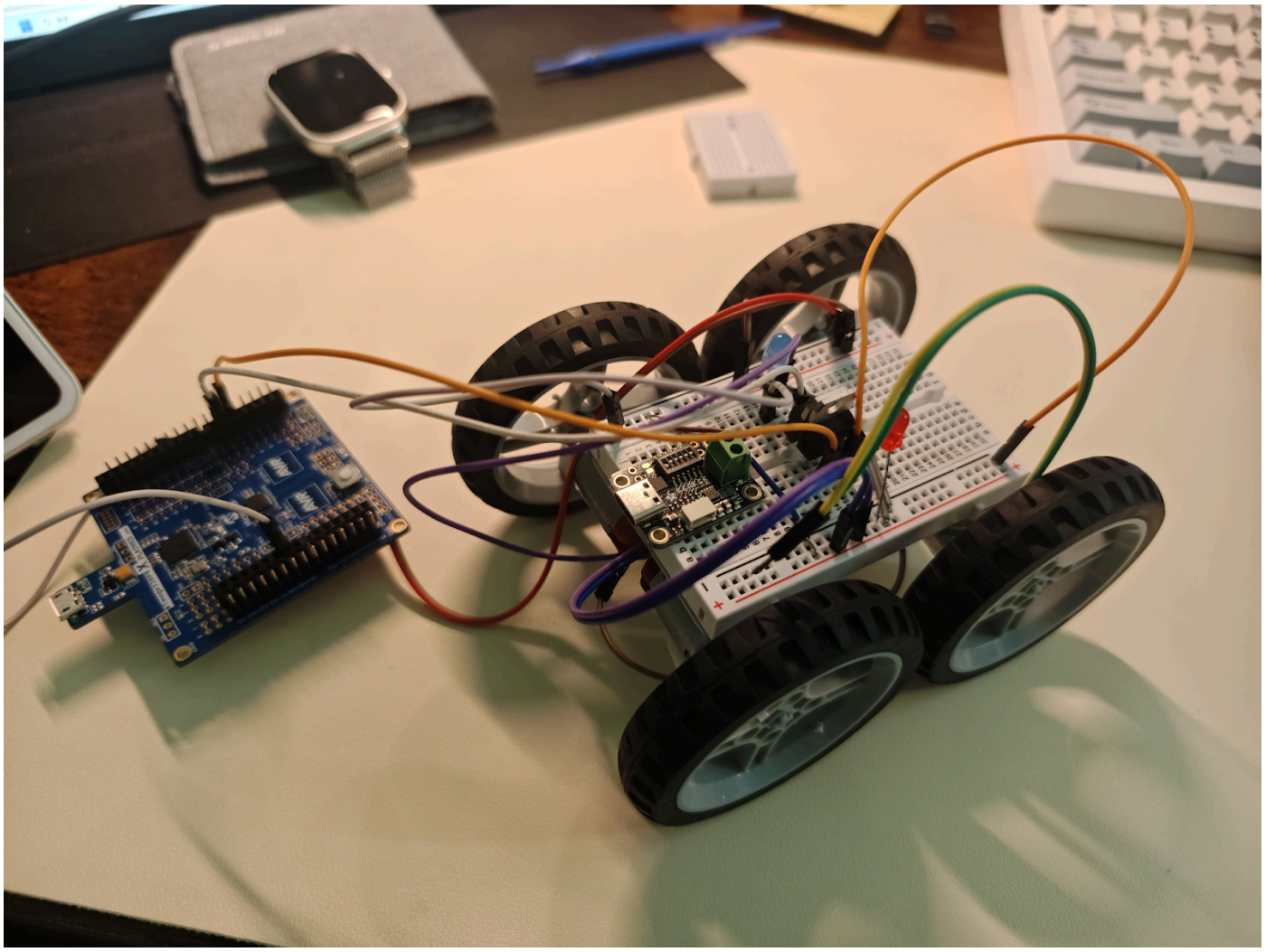
- ESP32 Firmware Logic
    - The ESP32 firmware integrates Wi-Fi connectivity, Blynk communication, and a UART command-forwarding engine. Incoming network commands are parsed and translated into compact structured frames that are transmitted to the ATmega328PB. This implementation maintains low communication latency and ensures consistent message delivery between cloud-based or PC-based control interfaces and the robot’ s local controller.

- ATmega328PB Firmware Logic
    - The ATmega firmware includes a UART parsing engine for interpreting commands forwarded by the ESP32, as well as a complete I^2C driver stack for interacting with the speech-recognition module. Both input channels feed a unified command dispatch layer responsible for triggering the appropriate motor-control routines. All communication routines are written in a non-blocking style to avoid interference with PWM timing, ensuring smooth and responsive motion output.

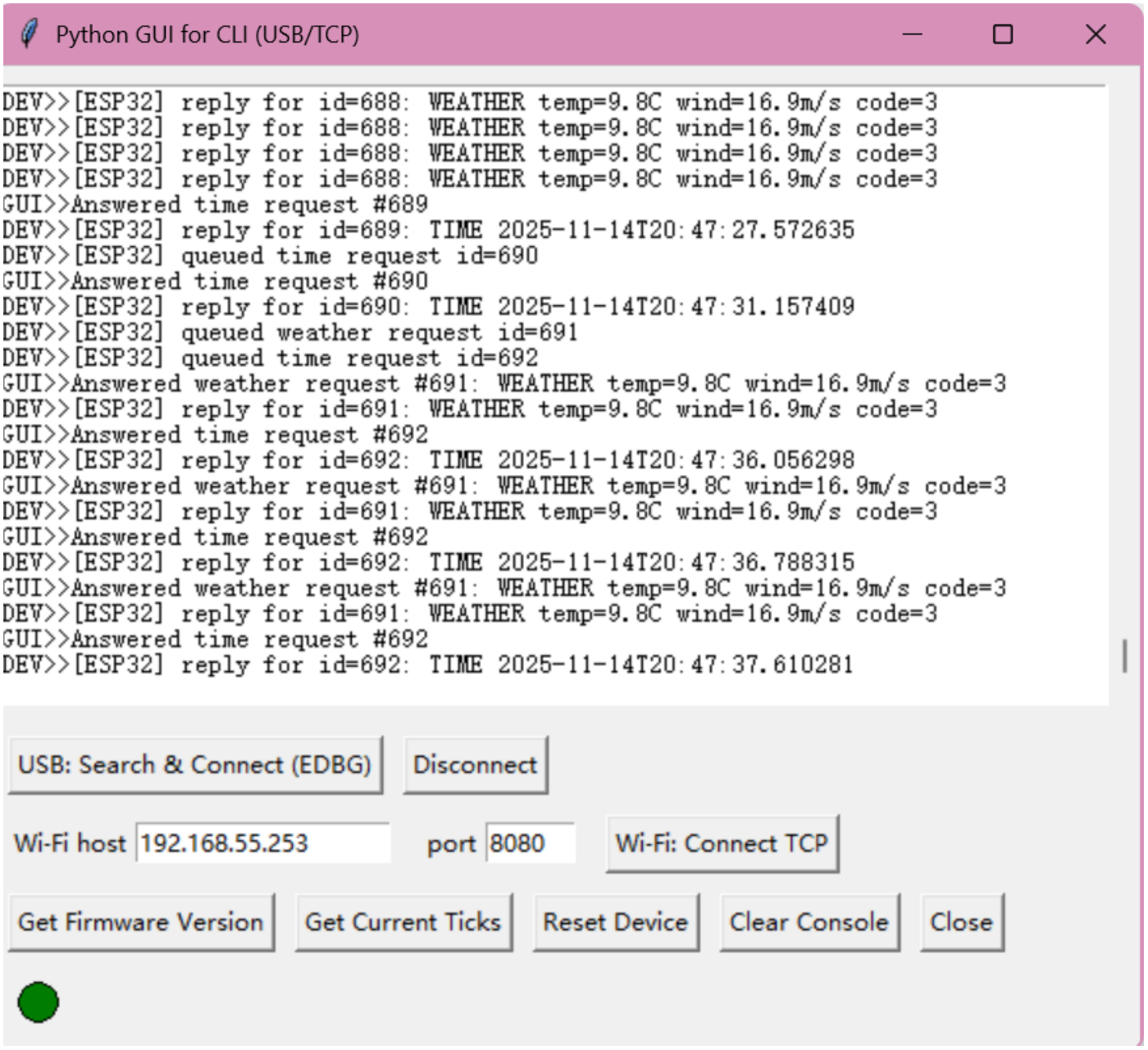
- Speech-Recognition Integration
    - Integration of the speech module is supported by a periodic detection loop that polls for new recognition results and processes the returned ID codes. A command-mapping mechanism translates these IDs into standardized motion commands understood by the main control layer. This ensures that voice commands and Wi-Fi commands share a unified control behavior and produce identical responses at the actuator level.

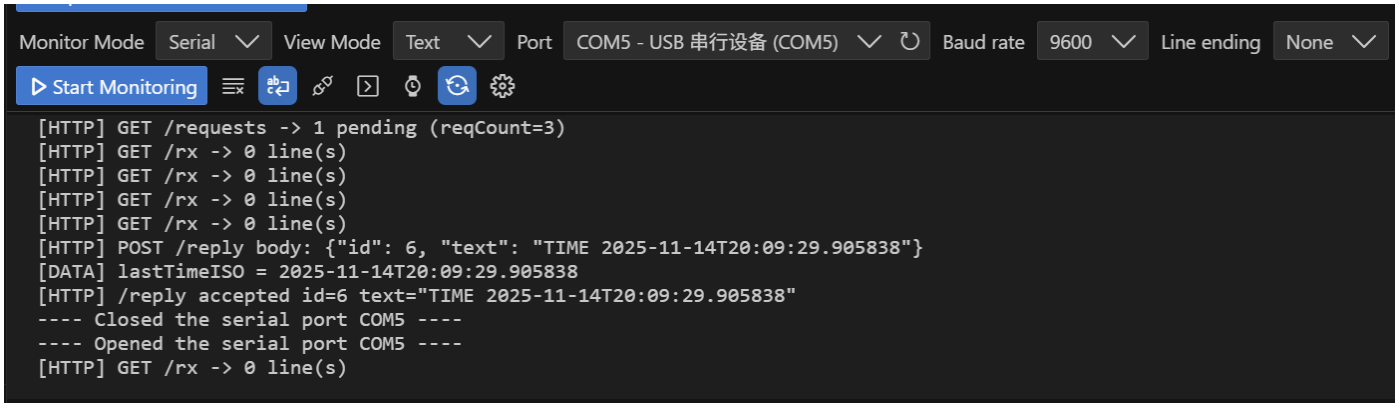
3. Demo your device.

Motion part:

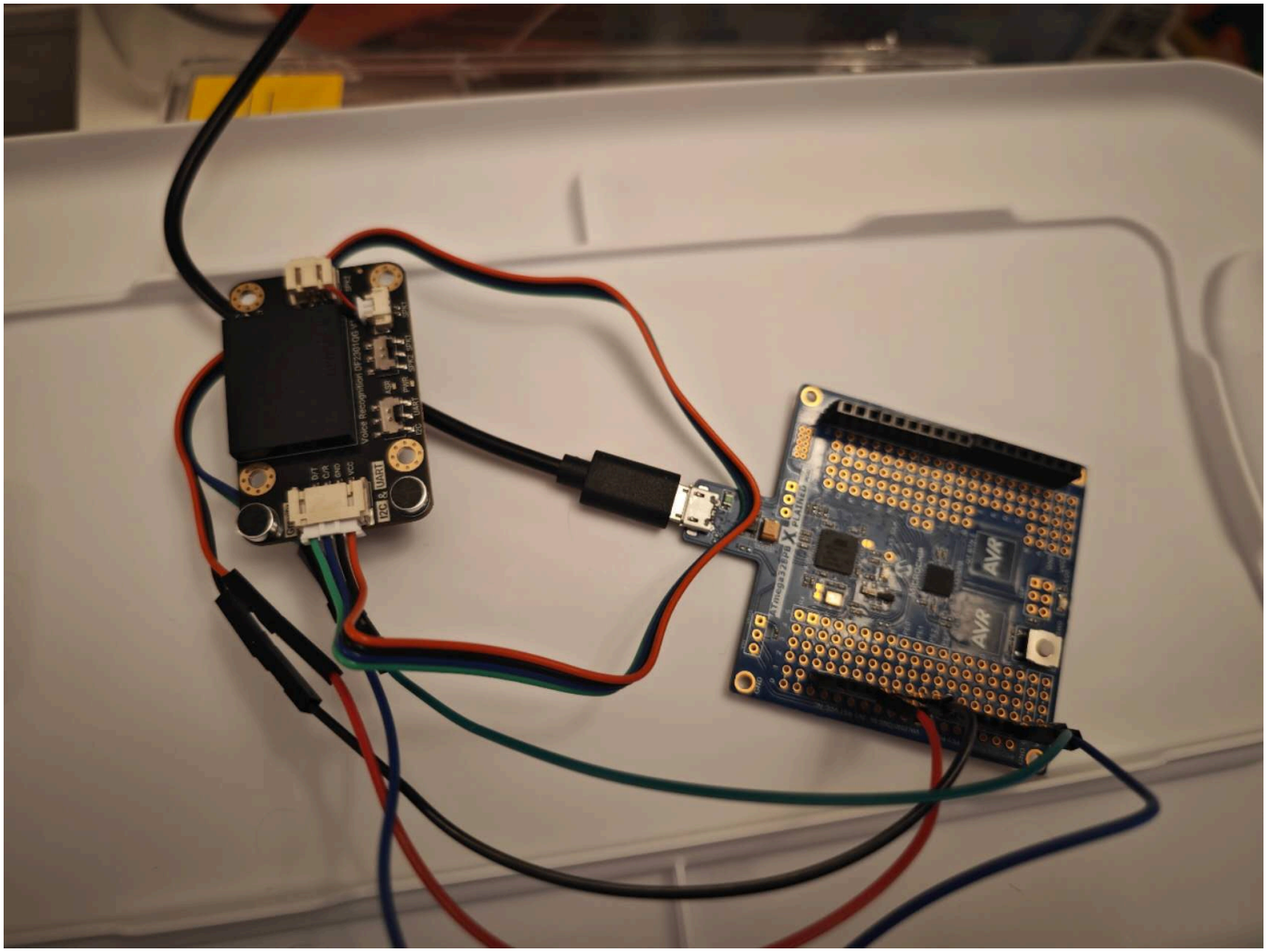


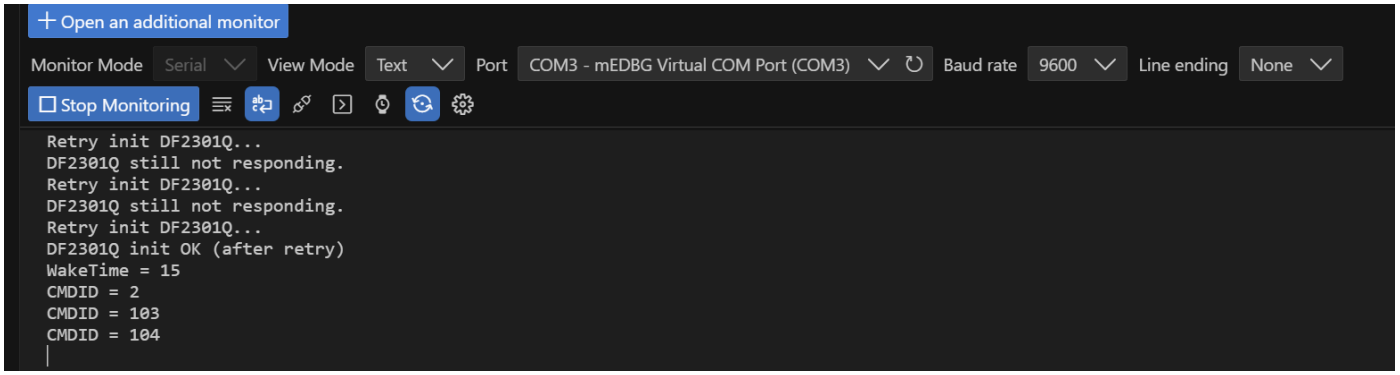
PC GUI:



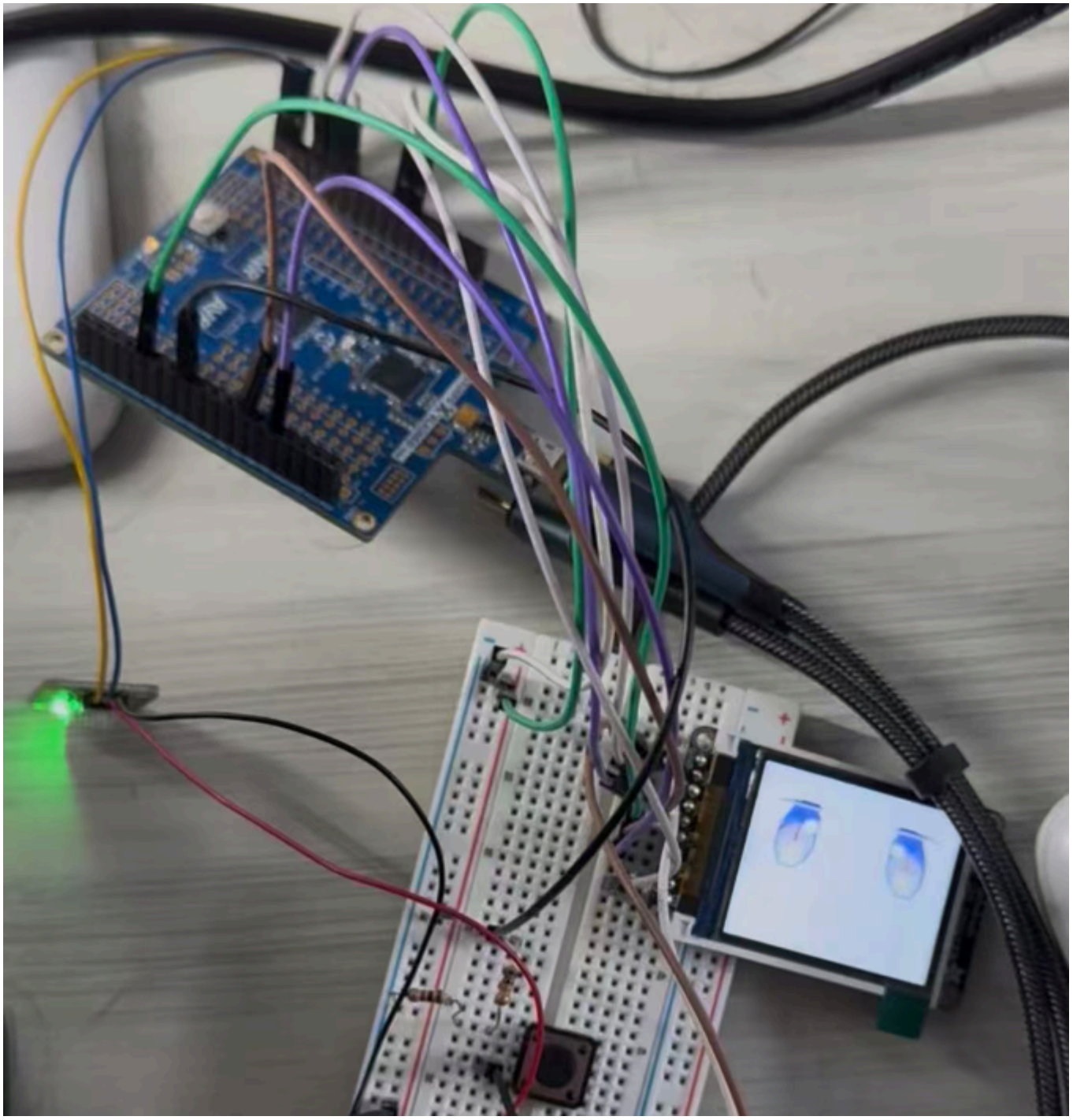


Voice recognition module:




Open an additional monitor
Monitor Mode
Serial
View Mode
Text
Port
COM3 - mEDBG Virtual COM Port (COM3)
Baud rate
9600
Line ending
None
Stop Monitoring
Retry init DF2301Q...
DF2301Q still not responding.
Retry init DF2301Q...
DF2301Q still not responding.
Retry init DF2301Q...
DF2301Q init OK (after retry)
WakeTime = 15
CMDID = 2
CMDID = 103
CMDID = 104

LCD screen:



4. Have you achieved some or all of your Software Requirements Specification (SRS)?

- The AHT20 temperature readings are correctly processed with 1 °C resolution, and humidity values are displayed with 1 % RH precision, meeting the required display accuracy.
- Our optimized LCD rendering pipeline allows partial-area refreshes to complete within ⩽ 50 ms, fully satisfying the timing requirement for smooth animation and UI updates.
- The DFPlayer Mini also meets its software-level responsiveness requirement, as it can change playback state quickly upon receiving UART commands.
- The servo driver meets the positioning accuracy requirements, successfully targeting specific angles to perform clear 'Yes' (nodding) and 'No' (shaking)

head gestures without jitter.
- The motor control system achieves rapid response times, enabling the robot to start,
- stop, and turn instantly upon command.
- The system successfully receives motion commands over Wi-Fi through the ESP32 and forwards them to the ATmega328PB via UART.
- The speech-recognition module reliably provides command IDs over I^2C after library porting and command-set training.
- A unified command-dispatch pipeline correctly maps both Wi-Fi and voice inputs to the same motion-control API.
- End-to-end command latency remains below the 1-second requirement for both communication pathways.
- UART and I^2C communication stacks operate in a fully non-blocking manner to ensure stable real-time control.

5. Have you achieved some or all of your Hardware Requirements Specification (HRS)?

- The AHT20 sensor is powered from a stable 5 V supply with ripple well below 50 mV peak-to-peak, and the wiring length is kept under 20 cm, routed away from high-dv/dt signals to prevent interference.
- The LCD subsystem operates correctly and can display temperature/humidity data, five distinct facial expressions, and text responses triggered by voice commands.
- For audio, the DFR0299 module meets the hardware timing requirement: after a valid voice-command trigger, it produces the first audible syllable within $\leqslant$ 1.5 s, and it is able to play ten distinct voice-response phrases, satisfying the functional specification.
- Locomotion Subsystem: The DC motors can provide consistent torque and stable rotational speed to ensure the robot chassis moves smoothly and fluidly across surfaces without stuttering.
- Servo Actuation: The servo motors are able to have a fast response time and high actuation speed to execute distinct "nodding" and "head-shaking" gestures immediately upon command.
- The ESP32 operates as a stable Wi-Fi interface and maintains reliable communication with external control sources.
- The UART connection between the ESP32 and ATmega328PB has been validated under high-frequency command transmission.

- The I^2C interface with the speech-recognition module consistently returns accurate recognition results.
- The ATmega328PB successfully converts received command packets into motor-control outputs through PWM/GPIO lines.
- Wireless control hardware and voice-control hardware are fully operational and integrated into the robot’ s actuation path.

6. Show off the remaining elements that will make your project whole: mechanical casework, supporting graphical user interface (GUI), web portal, etc.

- The remaining work mainly involves completing the physical enclosure and performing the final integration and assembly of all subsystems into the finished device.
- Several subsystems remain to be integrated to complete the full robotic platform. These include the TFT/LCD expression display, servo-based head-movement mechanisms, sensor modules (such as temperature or IMU sensors), and the final mechanical enclosure. In addition, a unified system-level controller must be implemented to merge wireless commands, voice commands, sensor inputs, animations, and motion routines into a single coordinated state machine. Completion of these components will allow seamless interaction between all subsystems and provide the final user-facing functionality envisioned for the complete design.

7. What is the riskiest part remaining of your project?How do you plan to de-risk this?

- The riskiest remaining aspect is that too many peripherals connected simultaneously may cause performance slowdown or latency. To de-risk this, we plan to optimize communication scheduling, reduce blocking operations, and ensure efficient wiring and power distribution so the system can run smoothly with all modules active.
- The most significant remaining risk involves multi-module integration on the ATmega328PB, where UART, I^2C, SPI, and PWM processes must coexist without timing conflicts or resource contention. Additional risk arises from command-source arbitration when voice and Wi-Fi inputs arrive simultaneously. These concerns will be mitigated through a non-blocking, event-driven firmware architecture, strict Timer allocation (e.g., Timer1 for servos, Timer0 for motors, Timer2 for periodic tasks), and the introduction of message-queue structures

for command serialization. A staged integration approach—beginning with communication subsystems, followed by actuator layers, and concluding with display and sensor modules—will reduce complexity and isolate potential faults early.

Final Project Report

Don't forget to make the GitHub pages public website! If you've never made a GitHub pages website before, you can follow this webpage (though, substitute your final project repository for the GitHub username one in the quickstart guide): https://docs.github.com/en/pages/quickstart

1. Video

[Insert final project video here]

- The video must demonstrate your key functionality.
- The video must be 5 minutes or less.
- Ensure your video link is accessible to the teaching team. Unlisted YouTube videos or Google Drive uploads with SEAS account access work well.
- Points will be removed if the audio quality is poor - say, if you filmed your video in a noisy electrical engineering lab.

2. Images

[Insert final project images here]

Include photos of your device from a few angles. If you have a casework, show both the exterior and interior (where the good EE bits are!).

3. Results

What were your results? Namely, what was the final solution/design to your problem?

3.1 Software Requirements Specification (SRS) Results

Based on your quantified system performance, comment on how you achieved or fell short of your expected requirements.

Did your requirements change? If so, why? Failing to meet a requirement is acceptable; understanding the reason why is critical!

Validate at least two requirements, showing how you tested and your proof of work (videos, images, logic analyzer/oscilloscope captures, etc.).

ID	Description	Validation Outcome
SRS-01	The IMU 3-axis acceleration will be measured with 16-bit depth every 100 milliseconds +/-10 milliseconds.	Confirmed, logged output from the MCU is saved to "validation" folder in GitHub repository.

3.2 Hardware Requirements Specification (HRS) Results

Based on your quantified system performance, comment on how you achieved or fell short of your expected requirements.

Did your requirements change? If so, why? Failing to meet a requirement is acceptable; understanding the reason why is critical!

Validate at least two requirements, showing how you tested and your proof of work (videos, images, logic analyzer/oscilloscope captures, etc.).

ID	Description	Validation Outcome
HRS-01	A distance sensor shall be used for obstacle detection. The sensor shall detect obstacles at a maximum distance of at least 10 cm.	Confirmed, sensed obstacles up to 15cm. Video in "validation" folder, shows tape measure and logged output to terminal.

4. Conclusion

Reflect on your project. Some questions to address:

- What did you learn from it?
- What went well?
- What accomplishments are you proud of?

- What did you learn/gain from this experience?
- Did you have to change your approach?
- What could have been done differently?
- Did you encounter obstacles that you didn’ t anticipate?
- What could be a next step for this project?

References

Fill in your references here as you work on your final project. Describe any libraries used here.