Final Presentation

Ubiquitous Lab Systems

CUBO

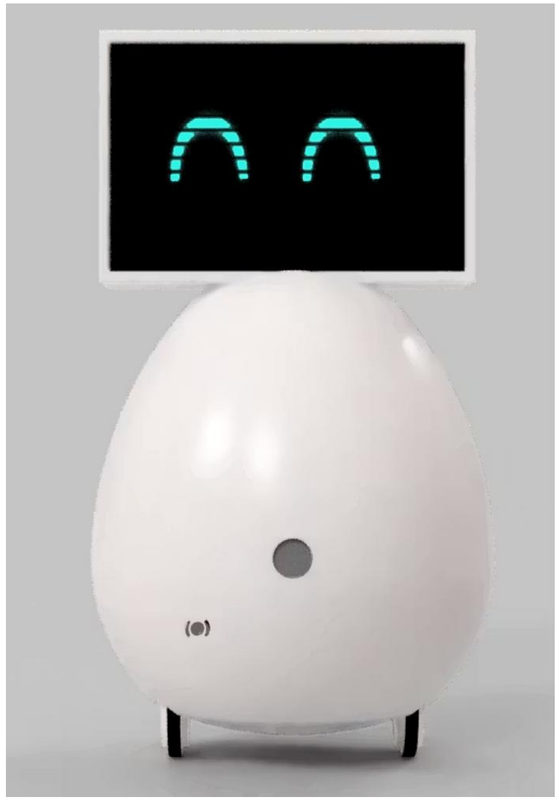
The Companion Robot for Modern Homes

Team :
Leena Kashyap – 1786572
Suryasaradhi Balarkan - 1840580

The Problem

- **Limited Interaction in Smart Devices**: Existing assistants like Alexa or Google Home are static and lack emotional or physical interaction.
- **Elderly Care Challenges**: Seniors often need assistance with reminders, emergencies, and companionship — which current devices do not offer.
- **Lack of Affordable Home Robots**: Most personal robots are expensive, complex, or require high technical expertise.
- **Fragmented Home Automation**: Many users struggle with integrating devices; no unified, intelligent controller.

Meet CUBO – The solution



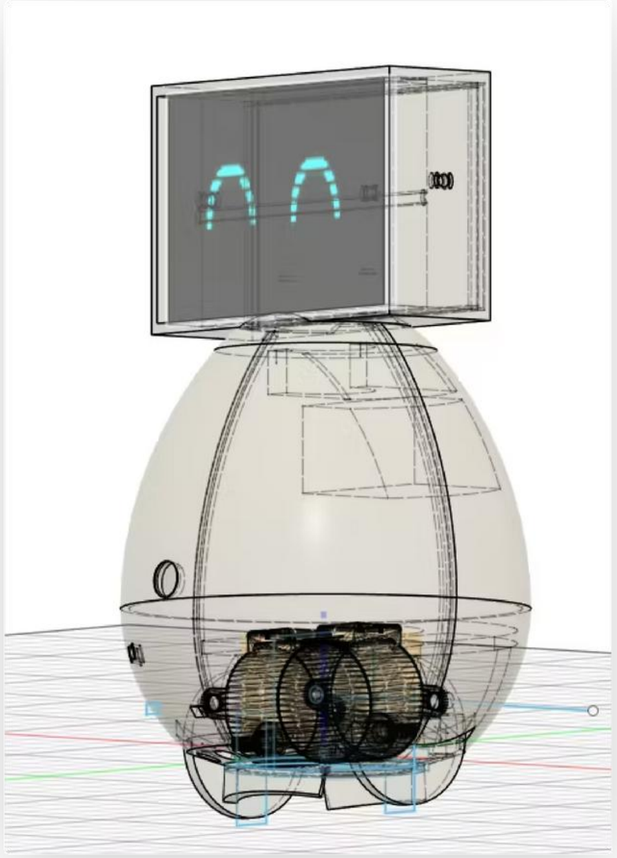
- CUBO = Alexa + Screen + Wheels + Feelings
- Fully autonomous, interactive, and helpful!
- The best thing: You can PET him! (PAT him) The robot automatically takes care of itself (charging & connection) as well as your family members schedules.
- CUBO integrates seamlessly into the family, needing NO ONGOING USER MANAGEMENT!

Opportunities

- **Rise of Affordable AI and Hardware**: Low-cost microcontrollers (RPi Zero 2W) and open-source AI tools make intelligent robots accessible.
- **Emotional Tech Demand**: Users want smart devices that feel "alive" — expressive, responsive, and relatable.
- **Expanding Market for Personal Robots**: Forecasts show rapid growth in domestic robotics for care, surveillance, and convenience.
- **Unified Platform Potential**: CUBO can serve as a central hub for home automation, personal scheduling, and smart communication.

Key Application

- Home automation and control
- Elderly care companion
- Educational assistant for children
- Ambient music, reminders, calendar management
- Mobile surveillance and security interface

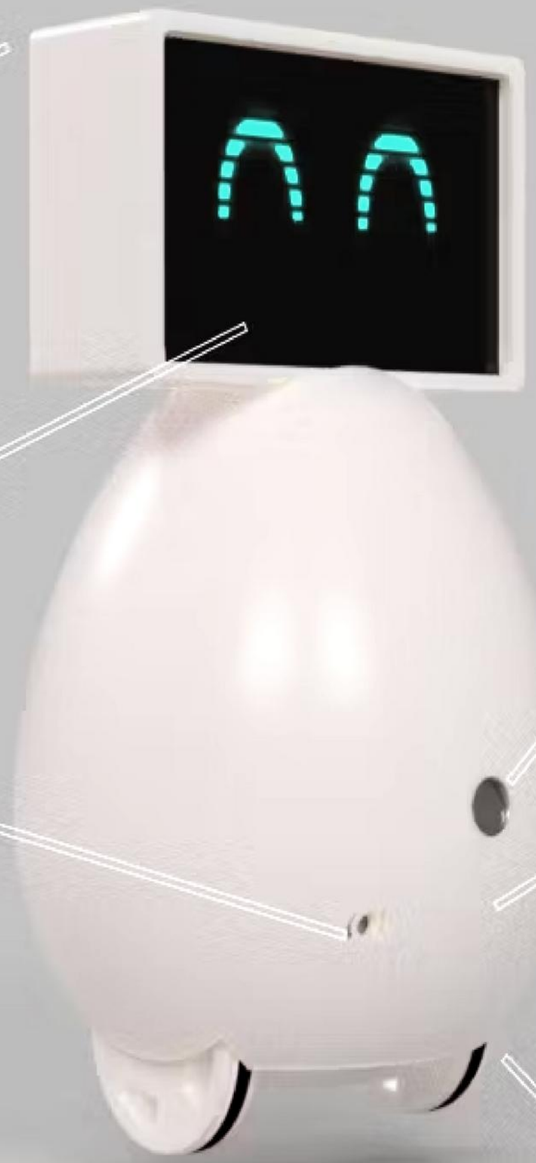


CUBO

HAPTIC Motion Detector
for you to PET Him

3 Inch Touch
Screen display

Condenser MIC for cubo to
never miss your talk



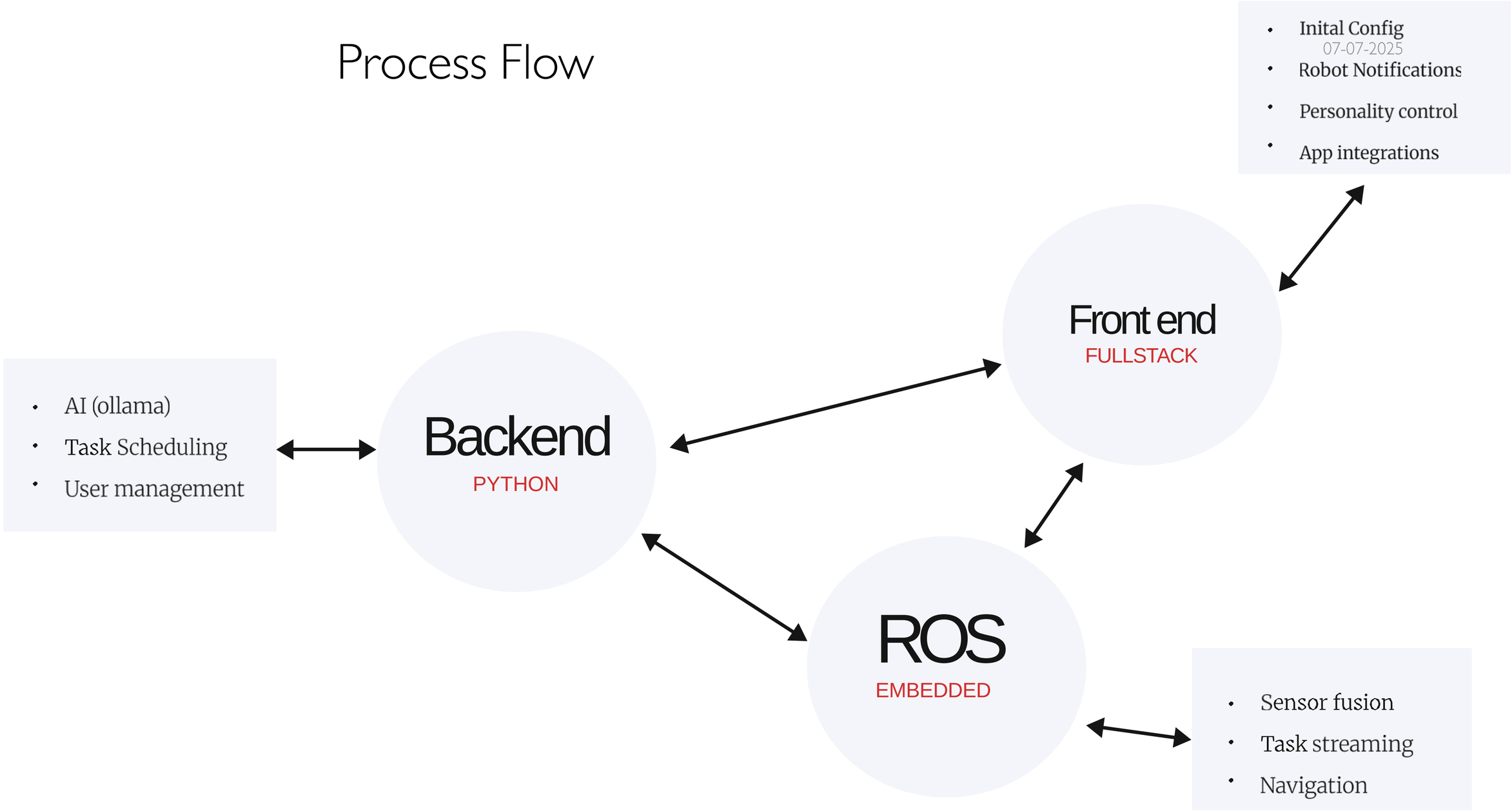
5MP Camera so that
cubo can see you

Raspberry PI driven
Opensource hardware

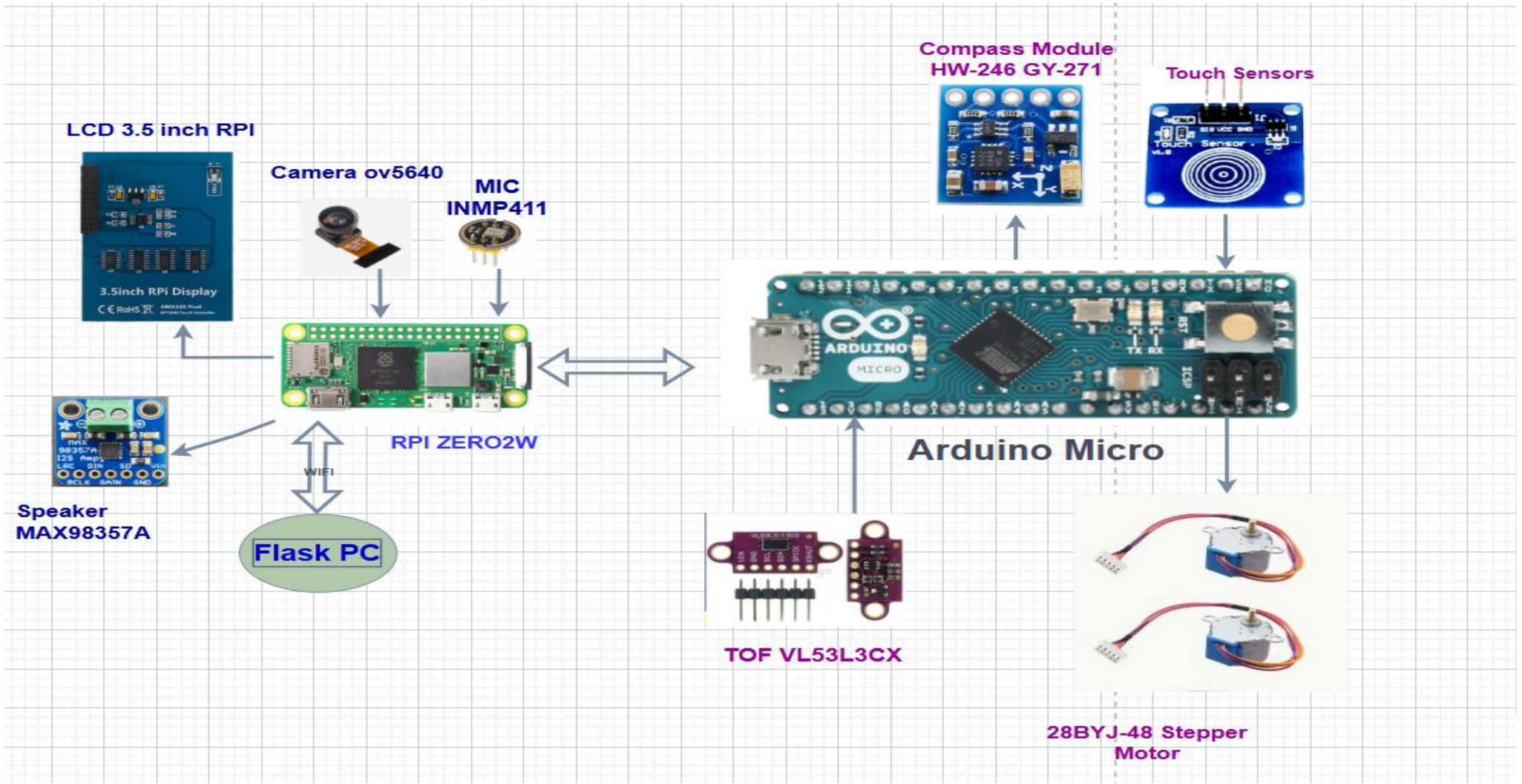
IR Edge detection

Hardware

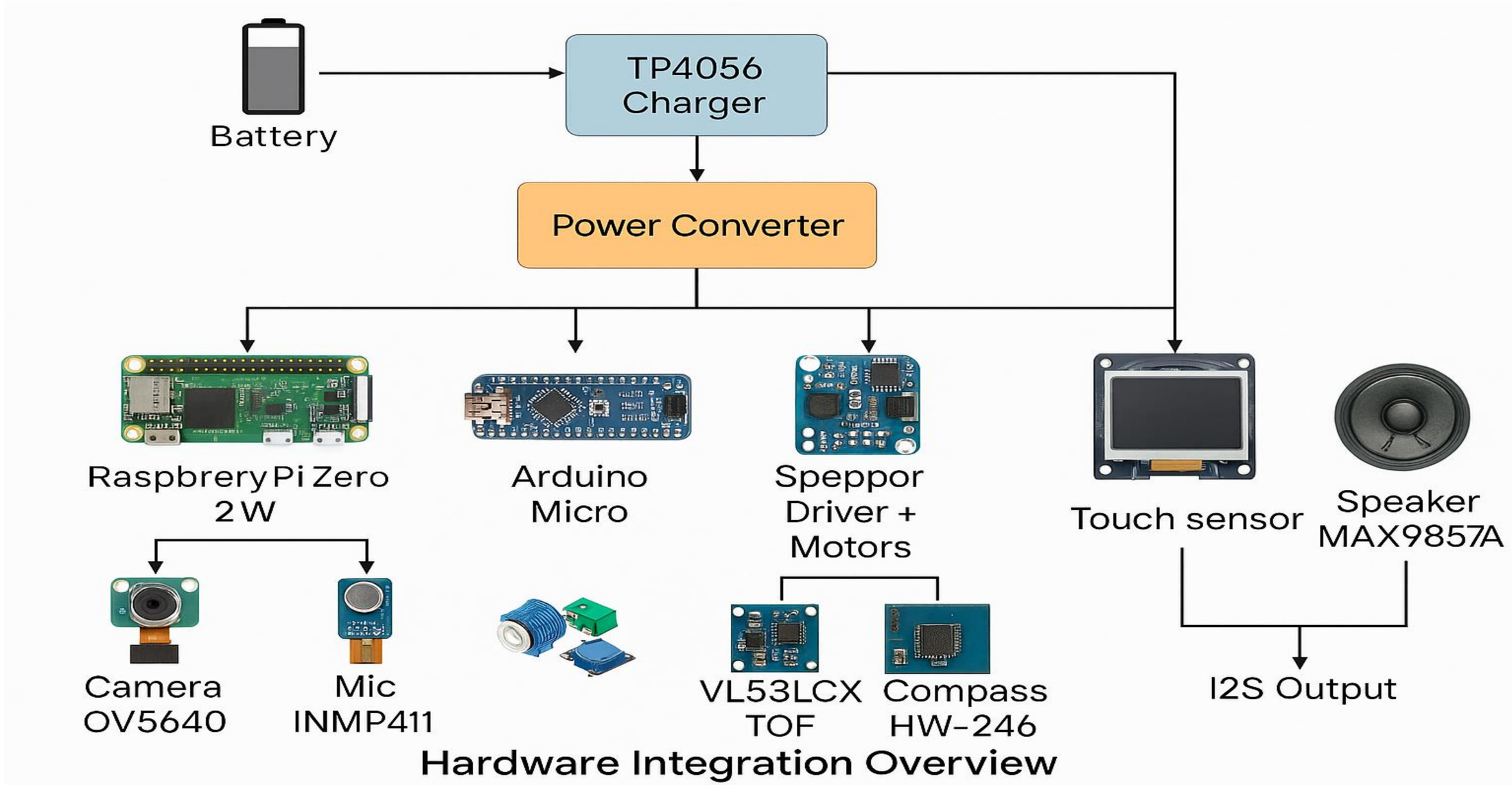
Process Flow



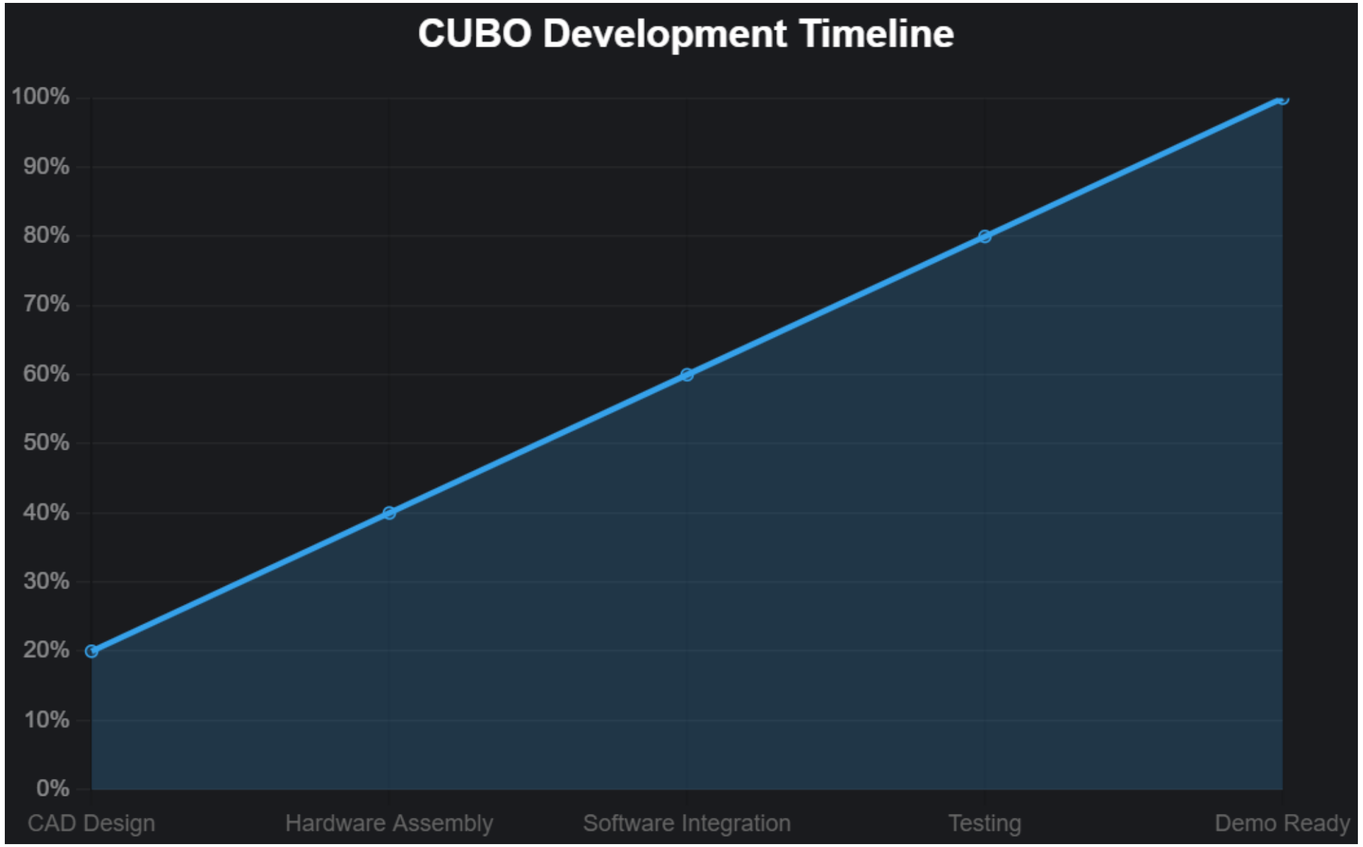
System overview



Hardware integration overview



Development Timeline



CAD Design - Multiple Revisions

Assembly Needed time

Sensors bought were different components (Wrote libraries from scratch)

Component Responsibilities in CUBO

Component	Function
TOF Sensor (VL53L3CX)	Measures distance to detect obstacles or motion
Compass Module (HW-246 GY-271)	Detects orientation (used for navigation or directional response)
Battery (3.7V Li-ion)	Main power source
TP4056 Charging Module	Safely charges the battery and protects against overcharge
Boost Converter (5V)	Converts battery output to stable 5V for RPi and other components
Flask PC (via WiFi)	Remote server for web interface and communication with CUBO

Component	Function
Raspberry Pi Zero 2W	Main processing unit (AI, camera, audio, UI, control logic)
Arduino Micro	Handles real-time control for motors and sensors
Camera (OV5640)	Captures video for object detection, streaming, and face tracking
Mic (INMP411)	Captures voice input for AI speech recognition (Whisper/VOSK)
3.5" LCD (SPI)	Displays UI elements like eyes, alerts, and feedback
Touch Sensor	Detects user interaction (petting, taps)
Speaker (MAX98357A)	Outputs audio (voice, responses, alerts) via I2S
Stepper Motors (28BYJ-48)	Moves robot parts like head or wheels
ULN2003 Driver Board	Drives stepper motors with power and direction control

Software stack

- C++,Cmake
- ROS – Sensor Control
- Python – Backend and AI
- ORABSLAM – Autonomous Navigation
- Fusion 360 – CAD modelling
- Ki-CAD – PCB design
- Flutter – Android App (Minimal)

Software Overview

Prompt based Actions
- Ollama Model
- Whisper Speech Recognition

Python Flask for Backend

WIFI

I2S Stream for Speaker Output

Camera Stream picamera

I2C Read from Lidar Module

U8glib
LCD Drawing directly from frame Buffer

PIGPIO
- ADC Direct Sampling
- GPIO Control for Stepper Motor

Charge control using ADC

Server (Linux)

Client (Raspibian)

Code Snippet : Compass Module

```
#include <Wire.h>
// code by surya for QMC5883P from datasheet https://www.qstcorp.com/upload/pdf/202202/%EF%BC%88%E5%B7%B2%E4%BC%A0%EF%BC%8913-52-19%20QMC5883P%20Datasheet%20Rev.C(1).pdf

#define compass_address  0x2C

void compass_WriteReg(byte Reg, byte val) {
  Wire.beginTransmission(compass_address);  //start talking
  Wire.write(Reg);                          // Tell the HMC5883 to Continuously Measure
  Wire.write(val);                          // Set the Register
  Wire.endTransmission();
}

byte compass_ReadReg(byte adrs) {
  Wire.beginTransmission(compass_address);  //start talking
  Wire.write(adrs);                         // Set the Register
  Wire.endTransmission();
  Wire.requestFrom(compass_address, 1);
  return Wire.read();
}

void compass_init() {
  compass_WriteReg(0x29, 0x06);  //Write Register 29H by 0x06 (Define the sign for X Y and Z axis)
  compass_WriteReg(0x0B, 0x08);  //Define Set/Reset mode, with Set/Reset On, Field Range 8Guass
  compass_WriteReg(0x0A, 0xCD);  //normal mode,odr = 200HZ
}


void compass_softReset() {
  compass_WriteReg(0x0B, 0x80);  // soft reset
  compass_WriteReg(0x29, 0x06);  //Write Register 29H by 0x06 (Define the sign for X Y and Z axis)
  compass_WriteReg(0x0B, 0x08);  //Define Set/Reset mode, with Set/Reset On, Field Range 8Guass
  compass_WriteReg(0x0A, 0xCD);  //normal mode,odr = 200HZ
}

/**
```

```
int compass_read_val(int* x, int* y, int* z) {

  delay(10);

  byte chip_id = compass_ReadReg(0x00);
  if (chip_id != 0x80) { return 0; }

  *x = (int)(int16_t)(compass_ReadReg(1) | (compass_ReadReg(2) << 8));
  *y = (int)(int16_t)(compass_ReadReg(3) | (compass_ReadReg(4) << 8));
  *z = (int)(int16_t)(compass_ReadReg(5) | (compass_ReadReg(6)<< 8));
  return 1;
}

float compass_azimuth(int* a, int* b) {
  float azimuth = atan2((int)*a, (int)*b) * 180.0 / PI;
  return azimuth < 0 ? 360 + azimuth : azimuth;
}

int compass_read(int* x, int* y, int* z, int* a) {
  int err = compass_read_val(x, y, z);
  *a = compass_azimuth(y, x);
  return err;
}

int x=10,y=11,z=12;
int az=5;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Wire.begin();
  compass_init();

}

void loop() {
  // put your main code here, to run repeatedly:
  delay(100);
  compass_read(&x,&y,&z,&az);
  Serial.print(x);
  Serial.print(" ");
  Serial.print(y);
  Serial.print(" ");
  Serial.print(z);
  Serial.println(" ");
}
```

Code Snippet : Stepper

```
//
// pin assignments, any digital pins can be used
//
const int LED_PIN = 13;

const int MOTORX_IN1_PIN = 4;
const int MOTORX_IN2_PIN = 5;
const int MOTORX_IN3_PIN = 6;
const int MOTORX_IN4_PIN = 7;

const int MOTORY_IN1_PIN = 8;
const int MOTORY_IN2_PIN = 9;
const int MOTORY_IN3_PIN = 10;
const int MOTORY_IN4_PIN = 11;


const int STEPS_PER_REVOLUTION = 2048;


//
// create two stepper motor objects, one for each motor
//
TinyStepper_28BYJ_48 stepperX;
TinyStepper_28BYJ_48 stepperY;



void setup()
{
  //
  // setup the LED pin and enable print statements
  //
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);


  //
  // connect and configure the stepper motors to their IO pins
  //
  stepperX.connectToPins(MOTORX_IN1_PIN, MOTORX_IN2_PIN, MOTORX_IN3_PIN, MOTORX_IN4_PIN);
  stepperY.connectToPins(MOTORY_IN1_PIN, MOTORY_IN2_PIN, MOTORY_IN3_PIN, MOTORY_IN4_PIN);
}


void loop()
{
  //
  // setup the speed, acceleration and number of steps to move for the
  // X motor, note: these commands do not start moving yet
  //
  stepperX.setSpeedInStepsPerSecond(300);
  stepperX.setAccelerationInStepsPerSecondPerSecond(500);
  stepperX.setupRelativeMoveInSteps(2048);


  //
  // setup the speed, acceleration and number of steps to move for the
  // Y motor
  //
  stepperY.setSpeedInStepsPerSecond(300);
  stepperY.setAccelerationInStepsPerSecondPerSecond(500);
  stepperY.setupRelativeMoveInSteps(-2048);


  //
  // now execute the moves, looping until both motors have finished
  //
  while((!stepperX.motionComplete()) || (!stepperY.motionComplete()))
  {
    stepperX.processMovement();
    stepperY.processMovement();
  }


  //
  // now that the rotations have finished, delay 1 second before starting
  // the next move
  //
  delay(1000);


  //
  // use the function below to move two motors with speed coordination
  // so that both stop at the same time, even if one moves farther than
  // the other
  //
  long stepsX = -2048 * 1;
  long stepsY = 2048 * 5;
  float speedInStepsPerSecond = 400;
  float accelerationInStepsPerSecondPerSecond = 1000;
  moveXYWithCoordination(stepsX, stepsY, speedInStepsPerSecond, accelerationInStepsPerSecondPerSecond);
  delay(3000);
}



//
// move both X & Y motors together in a coordinated way, such that they each
// start and stop at the same time, even if one motor moves a greater distance
//
void moveXYWithCoordination(long stepsX, long stepsY, float speedInStepsPerSecond, float accelerationInStepsPerSecondPerSecond)
{
  float speedInStepsPerSecond_X;
  float accelerationInStepsPerSecondPerSecond_X;
  float speedInStepsPerSecond_Y;
  float accelerationInStepsPerSecondPerSecond_Y;
  long absStepsX;
  long absStepsY;

  //
  // setup initial speed and acceleration values
  //
  speedInStepsPerSecond_X = speedInStepsPerSecond;
  accelerationInStepsPerSecondPerSecond_X = accelerationInStepsPerSecondPerSecond;

  speedInStepsPerSecond_Y = speedInStepsPerSecond;
  accelerationInStepsPerSecondPerSecond_Y = accelerationInStepsPerSecondPerSecond;
```

Code Snippet : TOF

```
#include <Wire.h>
#include <vl53lx_class.h>


#define DEV_I2C Wire
#define SerialPort Serial

#ifndef LED_BUILTIN
#define LED_BUILTIN 13
#endif
#define LedPin LED_BUILTIN

// Components.
VL53LX sensor_vl53lx_sat(&DEV_I2C, A1);


/* Setup ----------------------------------------------------------

void setup()
{
  // Led.
  pinMode(LedPin, OUTPUT);

  // Initialize serial for output.
  SerialPort.begin(115200);
  SerialPort.println("Starting...");

  // Initialize I2C bus.
  DEV_I2C.begin();

  // Configure VL53LX satellite component.
  sensor_vl53lx_sat.begin();

  // Switch off VL53LX satellite component.
  sensor_vl53lx_sat.VL53LX_Off();

  //Initialize VL53LX satellite component.
  sensor_vl53lx_sat.InitSensor(0x12);

  // Start Measurements
  sensor_vl53lx_sat.VL53LX_StartMeasurement();
}

void loop()
{
  VL53LX_MultiRangingData_t MultiRangingData;
  VL53LX_MultiRangingData_t *pMultiRangingData = &MultiRangingData;
  uint8_t NewDataReady = 0;
  int no_of_object_found = 0, j;
  char report[64];
  int status;

  do
  {
    status = sensor_vl53lx_sat.VL53LX_GetMeasurementDataReady(&NewDataReady);
  } while (!NewDataReady);

  //Led on
  digitalWrite(LedPin, HIGH);

  if((!status)&&(NewDataReady!=0))
  {
    status = sensor_vl53lx_sat.VL53LX_GetMultiRangingData(pMultiRangingData);
    no_of_object_found=pMultiRangingData->NumberOfObjectsFound;
    snprintf(report, sizeof(report), "VL53LX Satellite: Count=%d, #Objs=%1d ", pMultiRangingData->StreamCount, no_of_object_found);
    SerialPort.print(report);
    for(j=0;j<no_of_object_found;j++)
    {
      if(j!=0)SerialPort.print("\r\n                                ");
      SerialPort.print("status=");
      SerialPort.print(pMultiRangingData->RangeData[j].RangeStatus);
      SerialPort.print(", D=");
      SerialPort.print(pMultiRangingData->RangeData[j].RangeMilliMeter);
      SerialPort.print("mm");
      SerialPort.print(", Signal=");
```

```
    {
      if(j!=0)SerialPort.print("\r\n                                ");
      SerialPort.print("status=");
      SerialPort.print(pMultiRangingData->RangeData[j].RangeStatus);
      SerialPort.print(", D=");
      SerialPort.print(pMultiRangingData->RangeData[j].RangeMilliMeter);
      SerialPort.print("mm");
      SerialPort.print(", Signal=");
      SerialPort.print((float)pMultiRangingData->RangeData[j].SignalRateRtnMegaCps/65536.0);
      SerialPort.print(" Mcps, Ambient=");
      SerialPort.print((float)pMultiRangingData->RangeData[j].AmbientRateRtnMegaCps/65536.0);
      SerialPort.print(" Mcps");
    }
    SerialPort.println("");
    if (status==0)
    {
      status = sensor_vl53lx_sat.VL53LX_ClearInterruptAndStartMeasurement();
    }
  }

  digitalWrite(LedPin, LOW);
}
```

Code Snippet : UART bridge for Micro to Raspberry Communication

Code Snippet : Main code on Raspberry

```python
import serial
import time

from roboeyes import *
import time
from display_fbgen import Framebuffer


start_timer_val = time.perf_counter()  # high-precision timer

def ticks_ms():
        return (time.perf_counter() - start_timer_val) * 1000

lcd = Framebuffer()

# Start the display
lcd.fill(1)
time.sleep(1)


fps_screen = 7

# Open the serial port (adjust as needed)
ser = serial.Serial(port='/dev/ttyS0', baudrate=9600, timeout=1)  # Use '/dev/ttyUSB0' on Linux


# RoboEyes callback event
def robo_show( roboeyes ):
        lcd.update()

# Plug RoboEyes on any FrameBuffer descendant
robo = RoboEyes( lcd, 480, 320, frame_rate=fps_screen, on_show = robo_show )
```

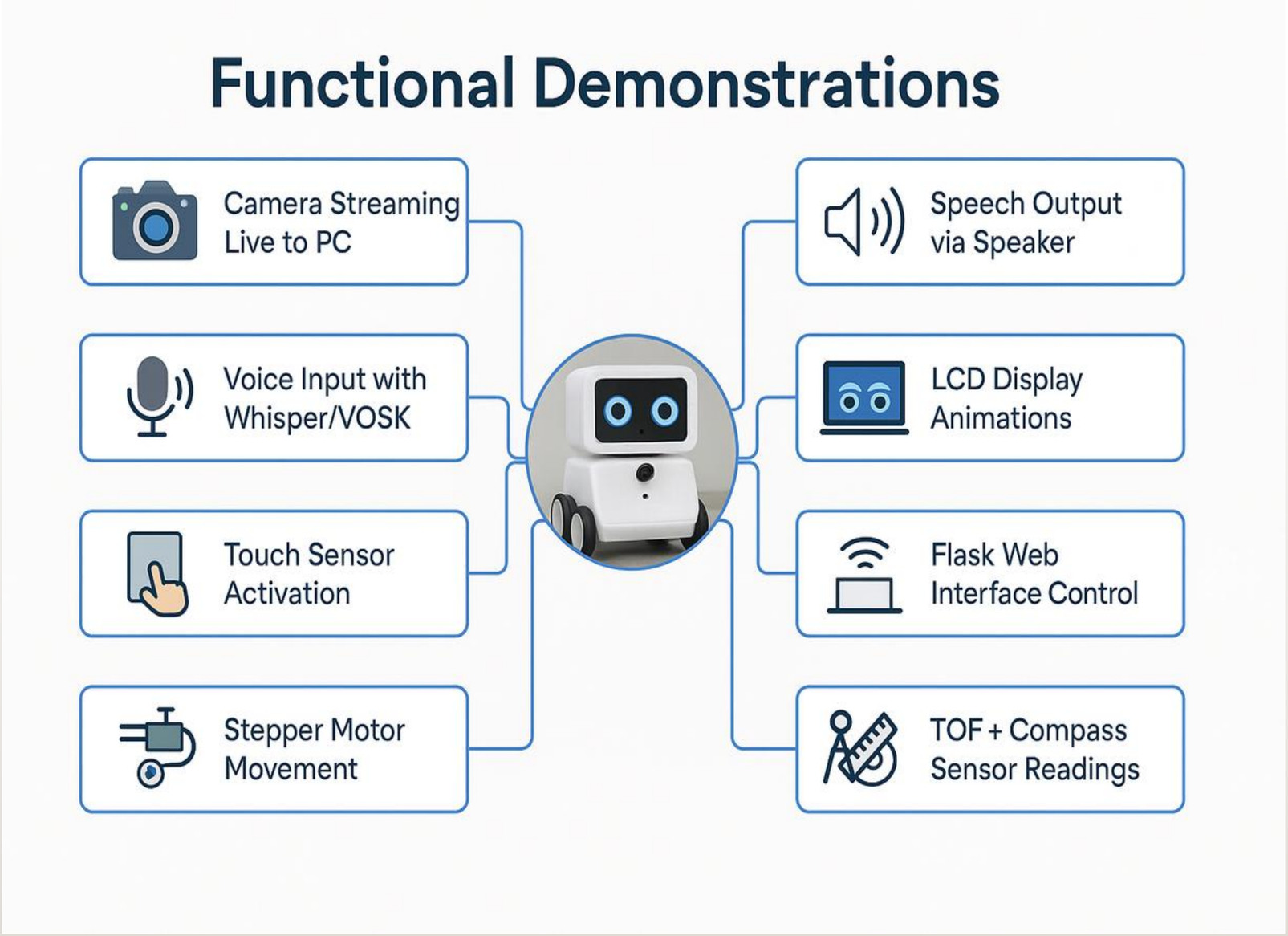
```python

# --[ Initial setup animation ]--
# Give a second to the eyes to open in their default state
start = ticks_ms()
while ticks_diff( ticks_ms(), start ) < 1000 :
        robo.update()

# --[ Open/Close Eyes ]--
# Auto blinker must be disable to properly run this
#robo.close() # Close Eyes
#robo.open() # Open Eyes

# --[ Define mood, curiosity and position ]--
robo.mood = DEFAULT  # mood expressions, can be TIRED, ANGRY, HAPPY, FROZEN, AFRAID, CURIOUS, DEFAULT
#robo.position = DEFAULT # cardinal directions, can be N, NE, E, SE, S, SW, W, NW, DEFAULT (default = horizontally and vertically centered)

robo.curious = True  # bool on/off -> when turned on, height of the outer eyes increases when moving to the very left or very right

# --[ Set horizontal or vertical flickering ]--
#robo.horiz_flicker(True, 2) # bool on/off, byte amplitude -> horizontal flicker: alternately displacing the eyes in the defined amplitude in pixels
#robo.vert_flicker(True, 2) # bool on/off, byte amplitude -> vertical flicker: alternately displacing the eyes in the defined amplitude in pixels

# --[ Play prebuilt oneshot animations ]--
#robo.confuse() # confused - eyes shaking left and right
#robo.laugh() # laughing - eyes shaking up and down
#robo.wink( right=True ) # make the right Eye Winking


# Send command
ser.write(b'sm;s1:-1000;s2:-1000\n')  # Send as bytes, add newline if Arduino expects it

```

```python
send_interval = 3  # seconds
last_send_time =  time.perf_counter()

try:
        while True:
                current_time = time.perf_counter()
                if current_time - last_send_time >= send_interval:
                        last_send_time = current_time
                        ser.reset_input_buffer()
                        ser.write(b'swa;\n')
                        time.sleep((1/fps_screen)/1.5)
                        # Check for incoming data
                        response = ser.readline().decode().strip()
                        if response:
                                if response.startswith("ACK;swa:"):
                                        value = int(response.split(":")[1])
                                        if (value > 9):
                                                robo.mood = HAPPY
                                                print("PAt Received,Robot happy")
                                        else:
                                                robo.mood = DEFAULT


                # update eyes drawings
                robo.update()
                time.sleep((1/fps_screen)/1.5) #set refresh rate slightly below fps of screen

except KeyboardInterrupt:
        print("Keyboard interrupt caught. Exiting gracefully...")
        ser.close()
except Exception as e:
        print(f"An error occurred: {e}")
        ser.close()
```

Implementation Process

- CAD model and robot frame assembled
- Hardware components connected and tested
- Python scripts for motors, camera, mic, and LCD
- Arduino microcontroller controls stepper motors
- Camera streaming enabled via Flask
- Touch and voice inputs working
- LCD shows robotic eye animations
- AI models integrated (Whisper, Ollama)
- Flask server running for remote control

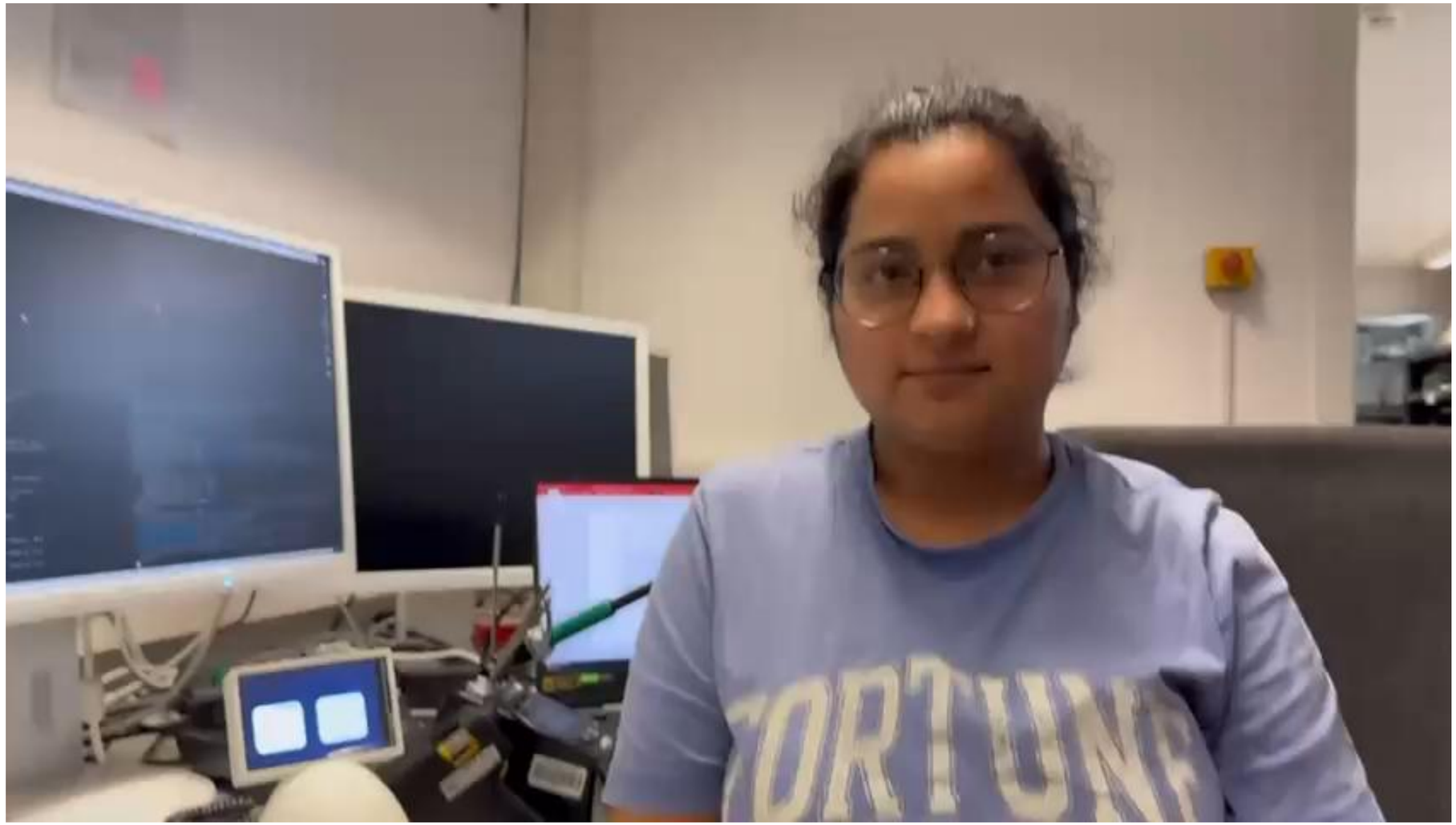
Functional Demonstration



Setbacks & Learning

- Delay in sourcing specific components
- Boost converter overheating under load
- Too many wiring
- Lack of components – We had to build even Level shifter
- Whisper model too slow for real-time use
- I2C/SPI bus conflicts during sensor integration
- Logic level mismatch between Arduino and sensors
- Learned to modularize and test each system
- Power planning is critical in mobile hardware
- Switched to VOSK for better real-time response
- Power consumption can cause system resets

Real use case – "CUBO is moving and talking"



Contributions and Future Scope

- https://github.com/thesunRider/cuBo

- Implement ORBSLAM
- Implement IOT backend
- Implement Whisper Model (switch from Vosk)
- ChatGPT Integration (Switch from Ollama)
- Add third party app integration (Skype,Whatsap)
- Connect to google IOT devices (Schedule alarms, Access calendars ..)
- Native language Speech Recognition
- PCB design and implementation
- Polishing Mechanical Design
- Faster Stepper motors
- Implement Gesture detection for user control

07-07-2025

Q&A