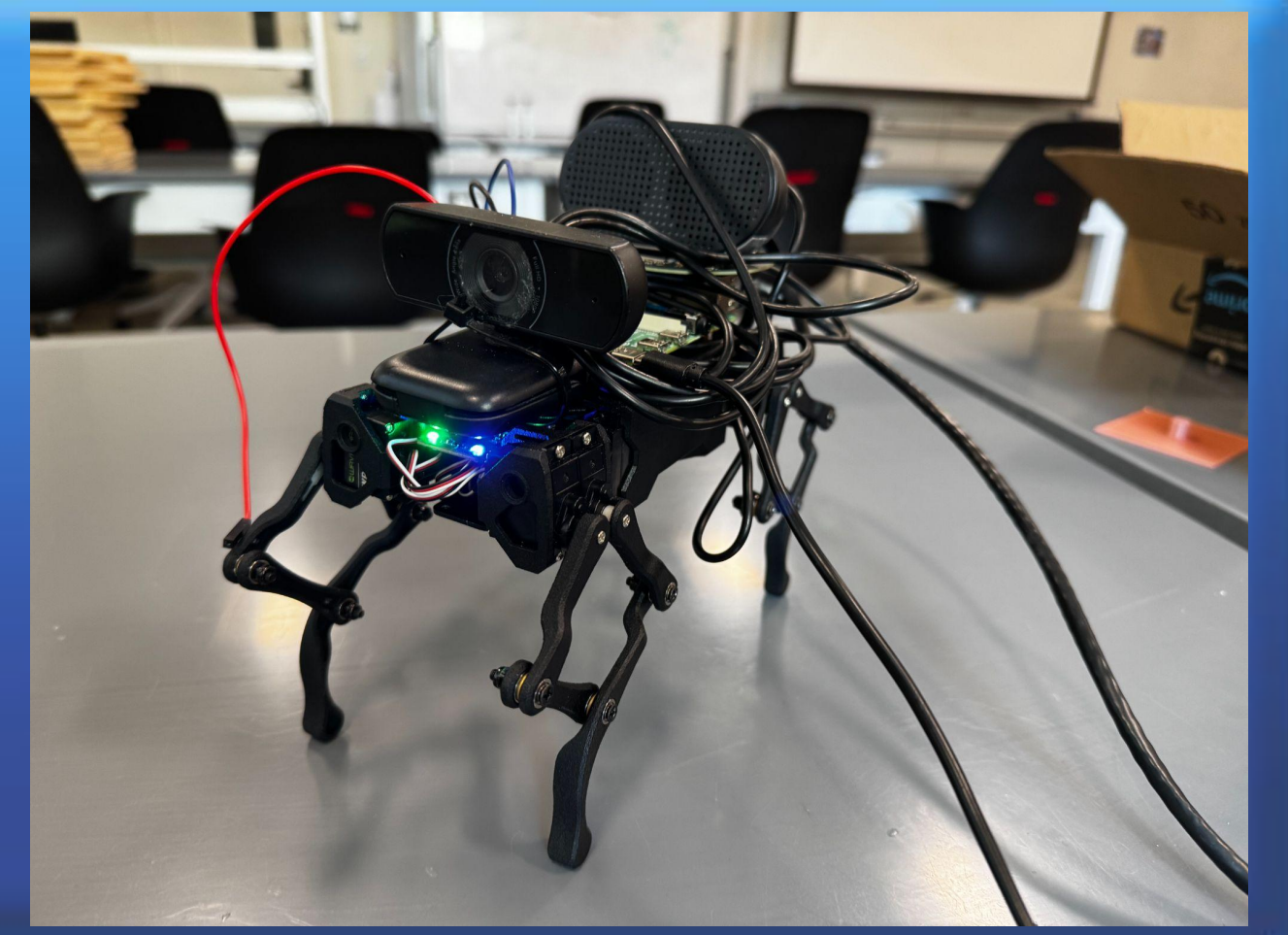




Design and Implementation of a Voice-Controlled Robot

Daniel Schaffield, Junxi (Steven) Li • Bahareh Abbasi • COMP 499



Introduction

As LLM's have grown more capable and tool calling standards have improved significantly, we sought out to see how accurate and useful an LLM controlled quad robot could be.

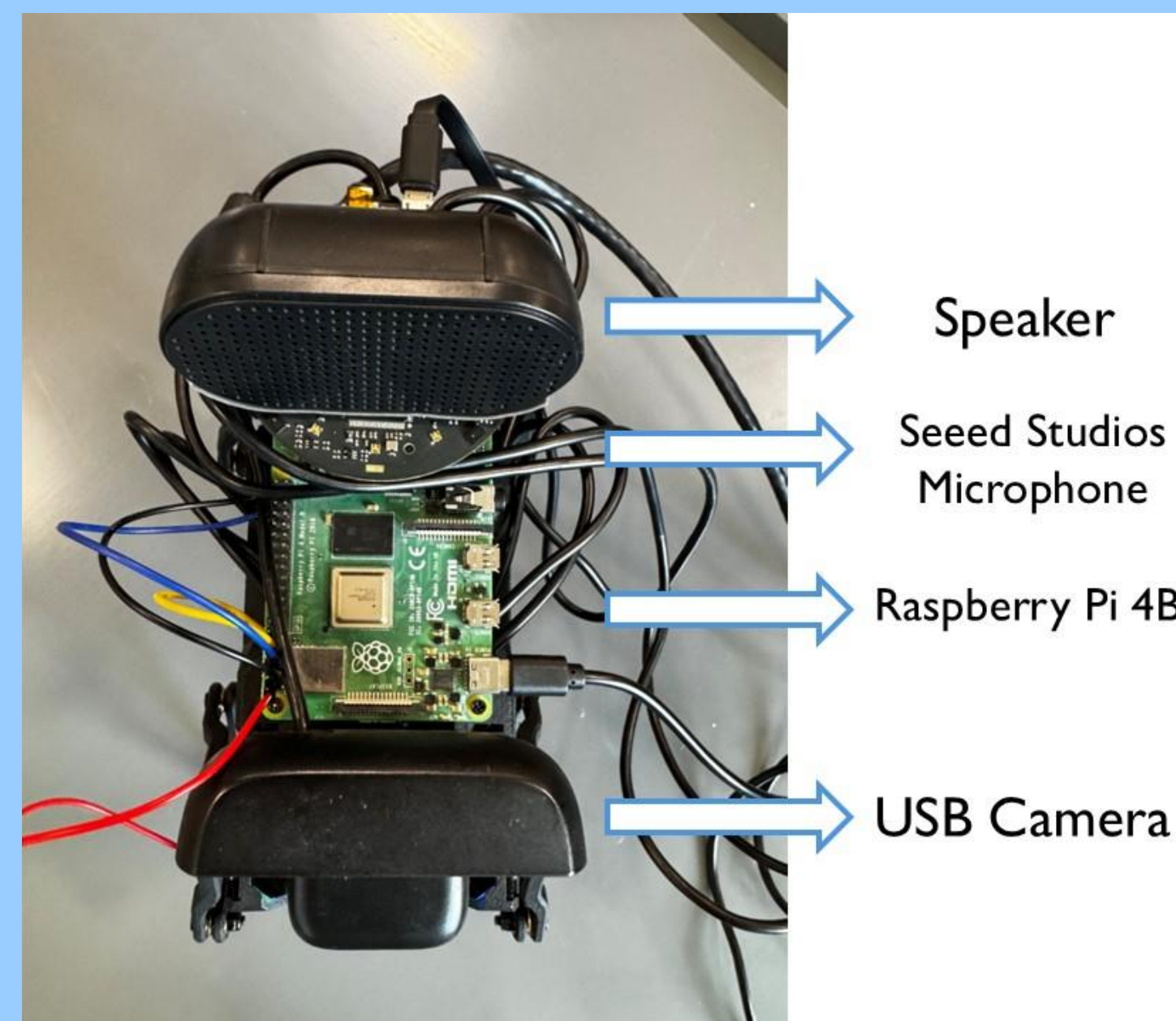
Our goal was to design a dog-like robot that can communicate to the user and bring entertainment by having physical movements and gestures.

Methodology and Components

The robot consists of several main key components:

- ESP32 Dev Module for robot control
- USB Camera
- Seeed Studios Microphone
- Raspberry Pi 4B
- Speaker

We used a WAVEGO Quadroped dog kit as our base, and strapped all the components to the bot as shown here:



When doing the programming, the ESP32 was primarily programmed in C / C++. Our Raspberry Pi robot script and serial connection is programmed in Python.

Features and Future Plans

Our robot is able to run voice recognition software, text to speech models, accept user commands, move and rotate, as well as answer the users questions in an entertaining way.

We had originally planned to add a gripper to our robot, but we ended up running out of time. In the future we would like to add an arm that can manipulate objects to increase its capabilities even more.

Robot Architecture and Algorithms

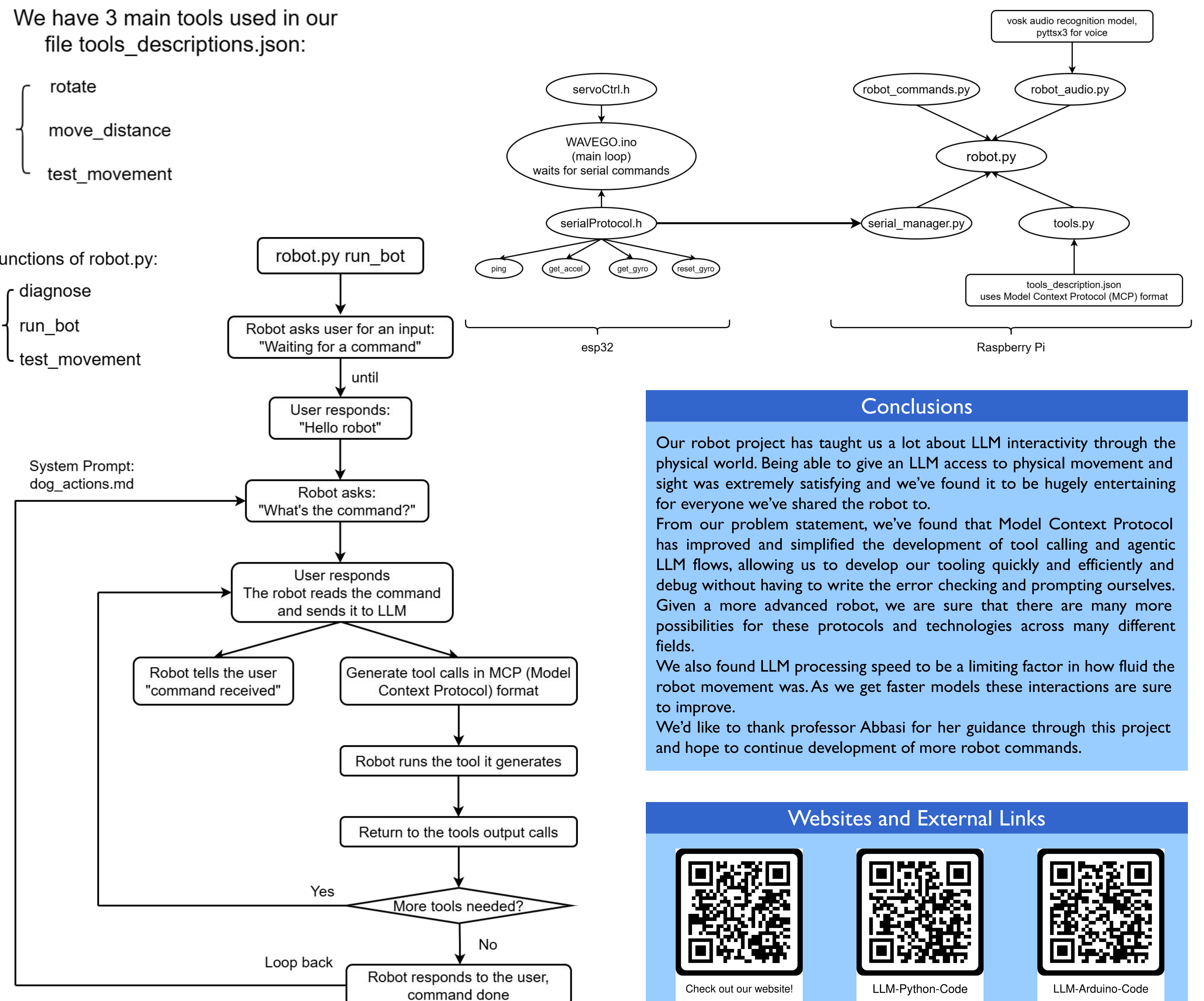
****Identity****: You are an autonomous quadruped robot dog with advanced locomotion and situational awareness.

We have 3 main tools used in our file tools_descriptions.json:

- rotate
- move_distance
- test_movement

3 functions of robot.py:

- diagnose
- run_bot
- test_movement



Conclusions

Our robot project has taught us a lot about LLM interactivity through the physical world. Being able to give an LLM access to physical movement and sight was extremely satisfying and we've found it to be hugely entertaining for everyone we've shared the robot to.

From our problem statement, we've found that Model Context Protocol has improved and simplified the development of tool calling and agentic LLM flows, allowing us to develop our tooling quickly and efficiently and debug without having to write the error checking and prompting ourselves. Given a more advanced robot, we are sure that there are many more possibilities for these protocols and technologies across many different fields.

We also found LLM processing speed to be a limiting factor in how fluid the robot movement was. As we get faster models these interactions are sure to improve.

We'd like to thank professor Abbasi for her guidance through this project and hope to continue development of more robot commands.

Websites and External Links



Check out our website!



LLM-Python-Code



LLM-Arduino-Code