Project Proposal: Boo-Bot

Daniel Garcia and Devin Pohl

CS 370

2/25/2021

Project Objective

The goal of this project is to create a small, remote controlled robot. This robot, nicknamed the "Boo-Bot" by the authors, will be capable of sending real-time video/audio and taking real-time controls over the internet, allowing it to be controlled from anywhere with a network connection. A specific goal is to allow this bot to be controlled via Twitch live-stream chat, yielding more participation during demos.

This project will be made as from-scratch as possible, with circuitry, drivers, and communications being designed and implemented throughout the course of the project wherever reasonable. We aim to make this as quality and clean as possible, leveraging the Electrical and Computer Engineering skills of the authors.

The primary board is currently planned as a Raspberry Pi 3 Model A+ running stock Raspbian Lite. Programming will be done in multiple languages (Rust, C, perhaps Python) and take advantage of inter-process communication as learned in class or via FFI bindings.

Members

Members of this project are as follows:

- Daniel Garcia
- Devin Pohl

Software Required

Overview

Software in this project can be sectioned into three main areas:

- Interacting with GPIO
- Streaming Video
- Interacting with Twitch Chat

Interacting with Twitch chat will be done through Rust. GPIO interaction will be done through Rust or C, whichever has better library support. Streaming video may be done through existing Unix utilities, likely `avconf`.

Justification

Rust will be used wherever possible. This is due to highly developed relevant libraries, increased safety over C, and better resource usage over C – we need every bit of performance possible. Rust's support for embedded and non-x86 environments has been rapidly maturing. Finally, Rust is just fun to program in, being much higher level than C or C++ without any of the penalties usually associated. If we find C to be better at a specific GPIO-related task, FFI can be used to provide slim bindings to Rust trivially easily.

Because the Raspberry Pi A+ is rather low-spec, video streaming will be one of the hardest challenges to overcome. As such, we currently plan to use existing utilities here which are already proven to be incredibly fast. Given that this task is much more common than the other two points, we expect this to be the best way forward. We suspect a combination of `ffmpeg` and `avconf` will be good options.

Hardware Required

Overview

Hardware in this project can be sectioned into 4 main areas

- SOC
- Sensors
- Actuators
- Power
- Miscellaneous

The SOC will be tasked with control over sensors and actuators, data processing, and wireless communication. Sensors retrieve environmental data and transmitted it to the SOC for further processing. Actuators will react under the command of the SOC. Power hardware supplies current to all the sensors, actuators, SOC, and miscellaneous hardware. Miscellaneous hardware are the support components; these may work as an intermediary between the SOC and the actuators.

Justification

The Raspberry pi will be the main data processing unit. The pi has the capability to process data at high speed and integration of wireless communication. The pi also includes many gpios and hardware accelerated data busses. Because of its outstanding capabilities, it is an adequate fit for the tasks in this project.

The sensors for this project will be the microphone, camera, ADC, and 9-axis MPU. These sensors will provide the user with the enviornmental data. The data can then be used to take desicions regarding future actions. The microphone will provide the user with audio input from its surroundings. The camera captures the visual feed from its surrounding. The ADC reads the battery level to prevent unexpected shutdowns. The 9-axis MPU provides feedback for collisions, speed, acceleration, direction, and rotation.

The actuators in this project allow the user to interact with its surroundings. These actuators are the display, speaker, servo, and stepper motors. The display shows text messages or visual cues sent by the server. The speaker allows the user to communicate back using audio. The server adjusts the tilt of the camera. The stepper motors enable the overall movement of the device.

Power is mainly focused on providing current to the entire system. The main components are the voltage regulators and the battery. The battery is the power storage and supply of this build. The regulators regulate the current provided by the battery before reaching all other electronic components.

Miscellaneous components are used in case-specific tasks. An example is data interpretation and amplification for speakers. Another example is data storage for the operating system. These hardware components are mostly used to support actuators, sensors, and the SOC.

Bill of Materials

Below is a bill of materials for this project, including all hardware used. All components are purchased from various EBay stores, with shipping times between one and three weeks.

Amount	Component	Price Ea ($)	Description	Cost ($)
2	Drivers	3.16	A4988 Stepper Motor Driver Module	6.33
2	Motors	3.98	MINEBEA NMB 2-phase 4-Wire 18° Stepper Motor	7.96
1	Amp	4.99	MAX98357A I2S Class D amplifier	4.99
1	Speaker	0.99	8 ohm speaker	0.99
1	Lipo	15.05	Lipo battery pack	15.05
1	Microphone	7.51	I2S MEMS Microphone SPH0645LM4H	7.51
1	Voltage Regulator	0.79	B628 3-24V to 12V 2A Adjustable Boost Step-Up Converter	0.79
10	Capacitors	0.466	16v 1000UF Electrolitic SMD	4.66
1	Raspberry pi	29.99	Raspberry Pi 3 Model A+ 2018 model	29.99
1	9-axis MPU	4.60	MPU9250 (Gyro, Accelerometer, Compass)	4.60
1	ADC	1.69	INA219 DC current and voltage sensor	1.69
1	Servo	1.79	SG90 9G Micro Servo Motor	1.79
1	Display	2.95	0.96“ I2C OLED Display	2.95
1	SD Card	5.00	32 GB Class 10 Micro SD Card	5.00
			Total:	94.30

Sketch

Below is a back-of-the-napkin CAD drawing. It shows the basic structure of the bot: a bit like a driving dome-style security camera.