The Lab Bot

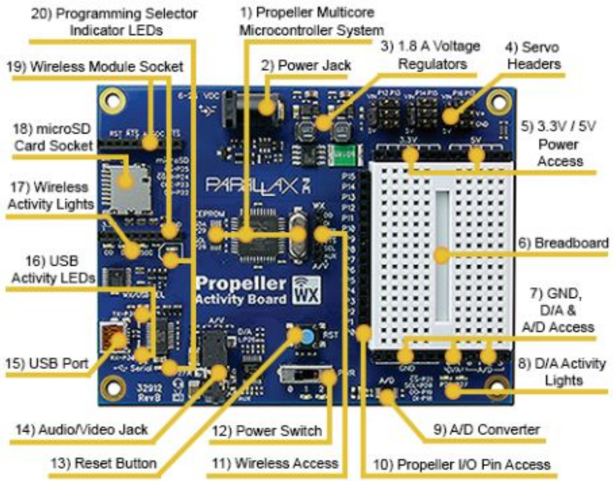
ARG2-D2
Nathan Labott Spring 2019

Definition and Requirements

- Definition
  - Design activities for a mobile robot that will hang out and "socialize" with students in the CS Lab.
- Requirements
  - Control the robot similar to that of an RC car
  - Move the bot to an object while avoiding obstacles
  - Several pre-programmed activities
  - Understand its environment and know the location of its contents
  - Handle racing conditions

Solution: Step 1 - Assembly

- Choice: Assemble or use a previous bot
- Disassembly (and then assemble again…)
- Bot orientation
- Test programs via USB port
  - Presented another problem (more of inconvenience)
- Along comes the WiFi module

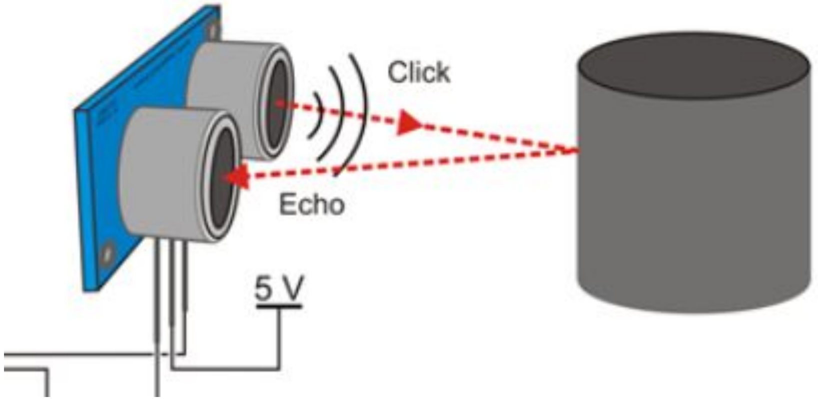

20) Programming Selector Indicator LEDs
1) Propeller Multicore Microcontroller System
3) 1.8 A Voltage Regulators
4) Servo Headers
19) Wireless Module Socket
2) Power Jack
18) microSD Card Socket
5) 3.3V / 5V Power Access
17) Wireless Activity Lights
PARALLAX
6) Breadboard
16) USB Activity LEDs
Propeller Activity Board WX
7) GND, D/A & A/D Access
15) USB Port
8) D/A Activity Lights
14) Audio/Video Jack
12) Power Switch
9) A/D Converter
13) Reset Button
11) Wireless Access
10) Propeller I/O Pin Access

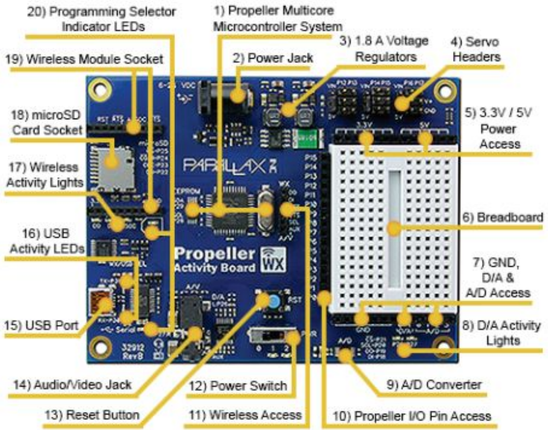
Step 2 - Wireless Capabilities

- Goal: everything to be wireless
- Wifi module acts as a sort of web server
    - Store files that interact with the microcontroller
- Allows all communication and control to happen via HTTP requests
- JavaScript functions (based on which HTML button is clicked) create HTTP requests that are then interpreted by the bot in C
- Bot additionally handles racing conditions quite well

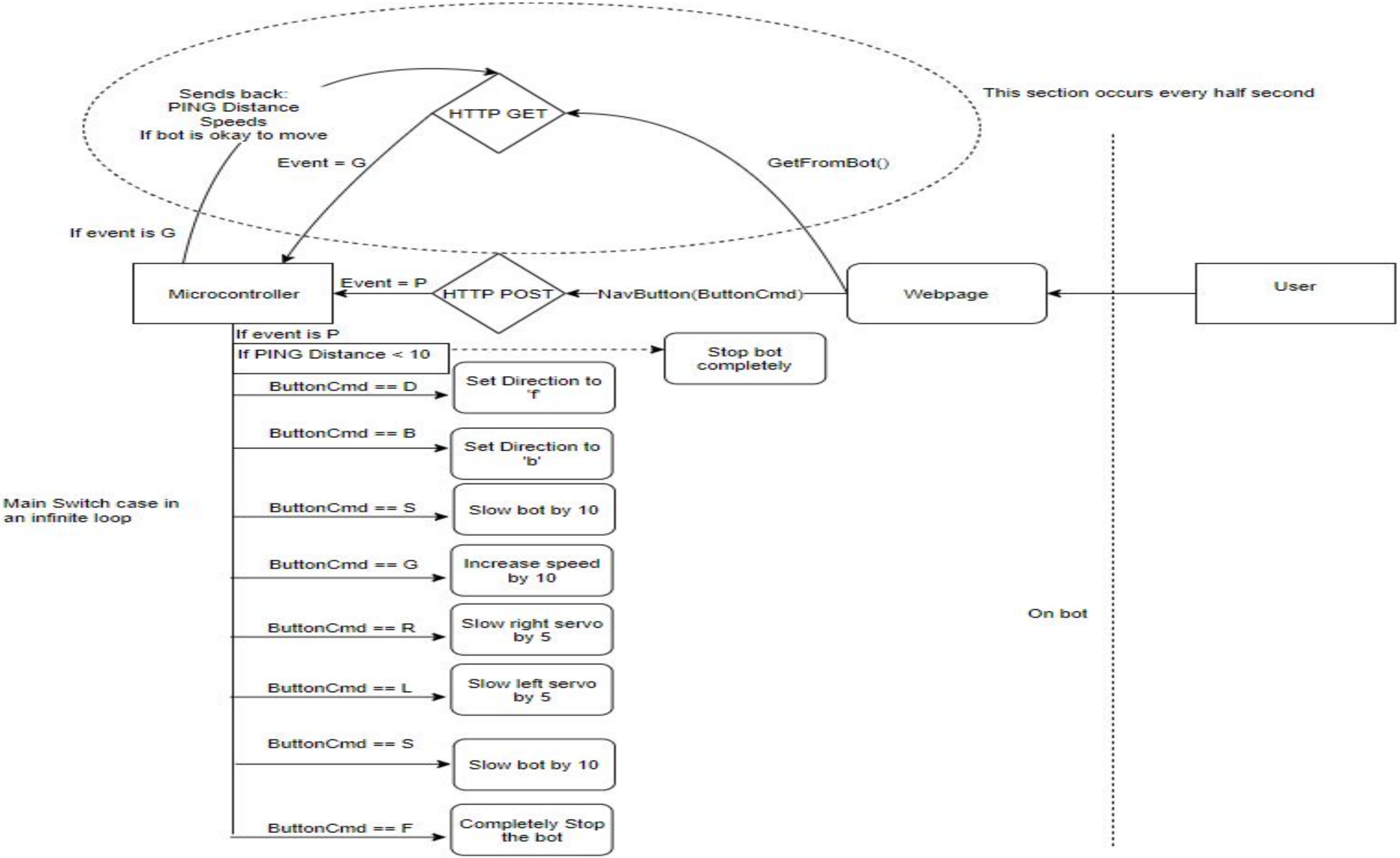
Step 3 - PING Sensor

- Ultrasonic sensor that detects distances
- How it works
  - Shoots out wave from one eye
  - Sets SIG pin to high
  - When wave comes back into other eye, sets SIG pin to low
  - Then measures how long the SIG pin was on high to determine distance
- Allowed for bot to maintain self-preservation
  - Limited on servo




20) Programming Selector Indicator LEDs
1) Propeller Multicore Microcontroller System
3) 1.8 A Voltage Regulators
4) Servo Headers
19) Wireless Module Socket
2) Power Jack
18) microSD Card Socket
5) 3.3V / 5V Power Access
17) Wireless Activity Lights
6) Breadboard
16) USB Activity LEDs
7) GND, D/A & A/D Access
15) USB Port
8) D/A Activity Lights
14) Audio/Video Jack
12) Power Switch
9) A/D Converter
13) Reset Button
11) Wireless Access
10) Propeller I/O Pin Access
PARALLAX
Propeller Activity Board WX

Demonstration


This section occurs every half second
Sends back:
PING Distance
Speeds
If bot is okay to move
HTTP GET
Event = G
GetFromBot()
If event is G
Microcontroller
Event = P
HTTP POST
NavButton(ButtonCmd)
Webpage
User
If event is P
If PING Distance < 10
Stop bot completely
ButtonCmd == D
Set Direction to 'f'
ButtonCmd == B
Set Direction to 'b'
Main Switch case in an infinite loop
ButtonCmd == S
Slow bot by 10
ButtonCmd == G
Increase speed by 10
ButtonCmd == R
Slow right servo by 5
ButtonCmd == L
Slow left servo by 5
ButtonCmd == S
Slow bot by 10
ButtonCmd == F
Completely Stop the bot
On bot

Exceptions

- Understands and knows the location of its contents
- Dancing (sort of)
- Audio - memory problems
  - Including more libraries took up too much space
  - Had to pick and choose which library was most vital

Methods and Techniques

- HTTP requests (POST and GET) created by JavaScript functions
- JavaScript Parsing (Split() function)
- Switch Cases
- Ports and a pipe of sorts (creates “pipe” on port 80 of wifi module)
- Assembly-like C that works with the microcontroller directly
- Hardware vs Software experiments

Learning and Dev Process

- Parallax Libraries on Github
- Parallax Learning Sites
- Parallax Community Forums
- A lot of, didn’t work - is it the software or hardware?

Extensions, What Could Be Added?

- Something with audio (Parallax plans to release smaller libraries in the near future - abdrive)
- Increased usage of the bots different cores (entire project just on one)

Questions, comments, Concerns?