Programmable Robot Teleoperation for Dummies

Remote control, monitoring, and analysis of robot behavior

Michael Tsang
September – December, 2014
Dr. Chen Li, Kaushik Jarayam, Dwight Swingthorpe
Professor Robert Full

Table of Contents

Overview

Robot Teleoperation

Remote control, monitoring, and analysis of robot behavior

Key Development:

Programmable teleoperation of a robot for any kind of robot actuation or even sensing.

Details:

The teleoperation includes both telemetry and telecommands between a computer and the robot's microcontroller. It can be applied to untether every lightweight cardboard-based robot used in lab.

Accessibility:

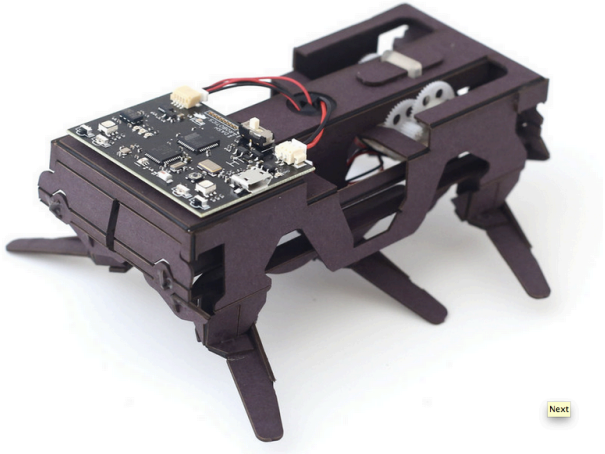
This powerful interface program I developed is user-friendly with instructions available at:
https://github.com/themichaeltsang/DashboardTeleoperation

Teleoperation: Telemetry + Telecommand

- Wireless link between computer and robot via Bluetooth-Low-Energy

Telemetry

Telecommand

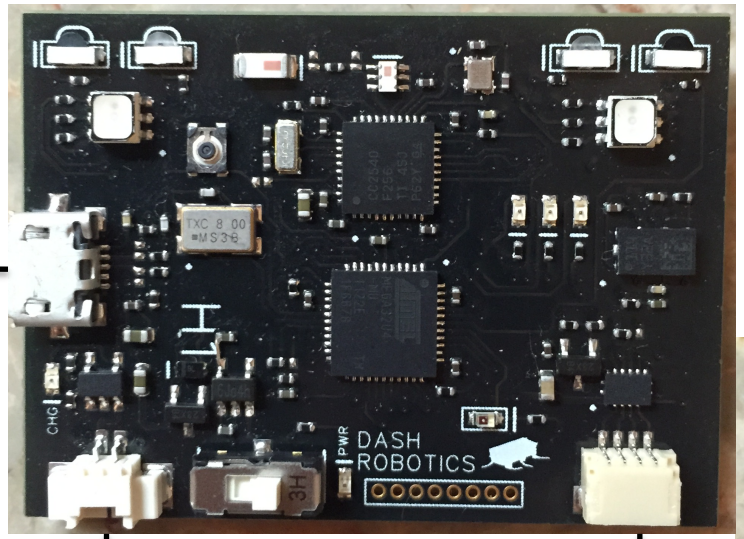


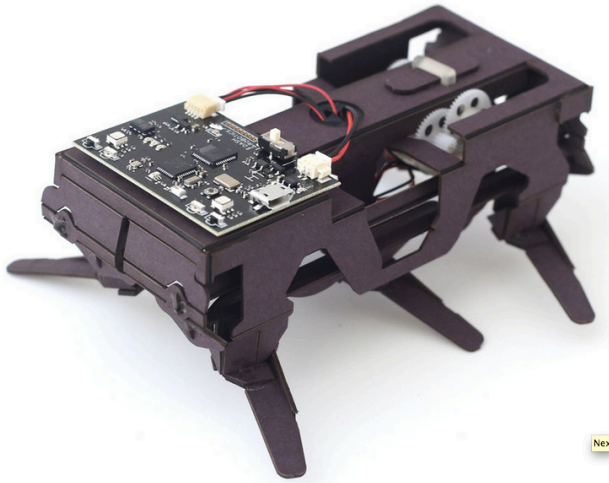
Mac OSX Supported, Requires Bluetooth 4.0

"Dashboard" Microcontroller Supported

Controller

Topview

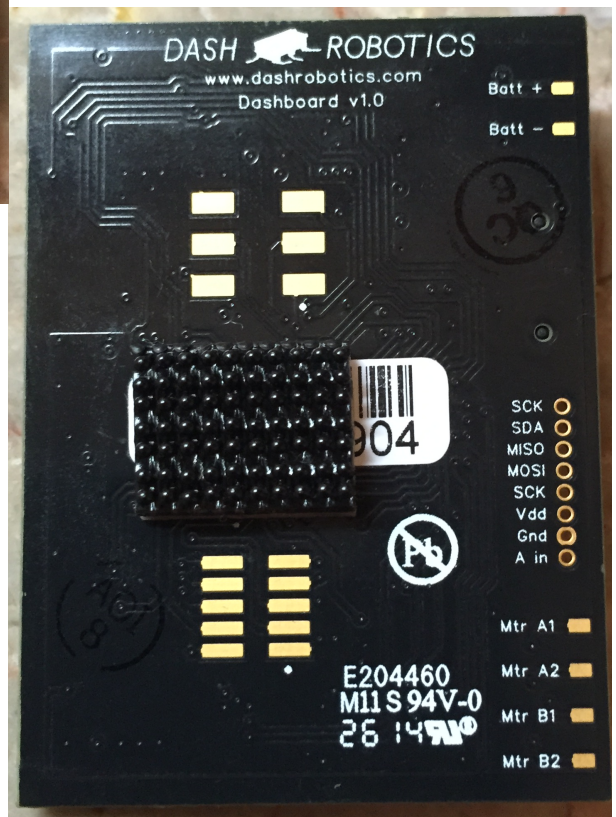




On-Off
Switch

Connect To Battery

Connect to dc
motors



"Expansion" Ports
See slide 32

Bottomview

Micro usb to computer for
Arduino programming

Usage should be as easy as…

Set Keypress Definitions

```
define( 'Start_Automation1_KEY', 'p' );
define( 'Start_Automation2_KEY', 'o' );
...
define( 'Run_Forward_KEY', 'a' );
define( 'Turn_R_KEY, 'd' );
define( 'Turn_L_KEY, 'a' );
...
define( 'Set_Servo1_Init_Position_KEY', 'z' );
define( 'Set_Servo1_Final_Position_KEY', 'x' );
...
```

Set Telemetry Parameters

```
define( 'Save_Sensor_Data', true );
define( 'Rate_of_Sensor_Data_Arrival', 100 ); //in Hz
```

Set Telecommand Parameters

```
define( 'Forward_Speed', 80 ); //unitless 1-100
define( 'R_Turn_Angular_Velocity', 70 ); //unitless 1-100
define( 'L_Turn_Angular_Velocity', 70 );
define( 'Turn_Duration', 300 ); //(ms)
...
```

After setting PARAMs,

Type into terminal:

`node main.js`

…and that's it!

Features: Telemetry

- Read sensor data for programs you write (see page 27 for code)
- Automatically save live sensor data sent wirelessly from robot onto computer

Sensor Data: Sat Nov 15 2014 21:26:27 GMT-0800 (PST)

Expected Time (ms)	Actual Time (ms)	Yaw Rate (deg/sec)	Ambient Light	L Proximity	R Proximity	L Motor(%)	R Motor (%)
0	65	8	21	709	675	0	0
100	133	4	20	710	677	0	0
200	134	0	20	710	677	0	0
300	200	8	21	706	674	0	0
400	269	4	22	703	671	0	0
500	336	4	24	700	670	0	0
600	404	8	24	698	668	0	0
700	404	0	23	699	669	0	0
800	470	4	22	699	669	0	0
900	538	8	21	701	671	0	0
1000	606	13	22	697	667	0	0
1100	606	0	24	694	663	0	0
1200	740	-4	26	689	662	0	0
1300	808	-13	26	692	664	0	0
1400	943	-17	23	696	668	0	0

Data saved in excel (csv) format on computer

Sensor data will be appended to file sensorlog.csv with a timestamp header

Features: Telecommand

- Keypress Move Forwards, Right, Left (Slide 30 for code)
- Keypress Actuate Servo 1, 2, or 3 (Slide 33)
- Keypress to start custom teleoperation program you define
  - Automated control over dc motors, servos via telecommand (Slides 31, 34)
  - Simultaneous control of motors (Slide 36)
  - Access to sensor readings via telemetry (Slide 27)

Installation

Installation

1. Setup Microcontroller
2. Setup Computer

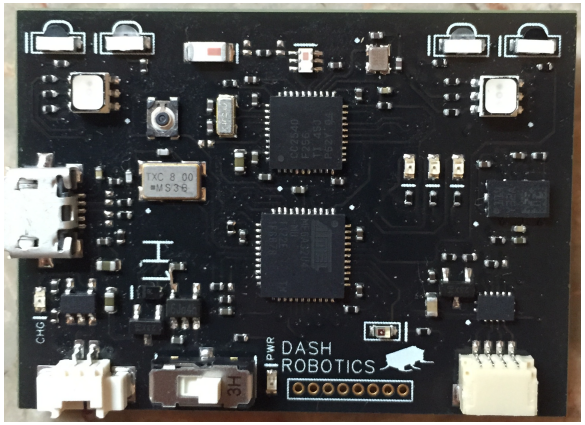
Setup Microcontroller

1. Load Custom Arduino Firmware

Visit:
https://github.com/themichaeltsang/DashboardTeleoperation

Follow these two points closely (copied from the link above)

- Learn how to customize DashBoard firmware
- Replace the folder "/libraries/DashBot" … with the "/libraries" folder, found inside "custom_arduino_files" in [the Dashboard Teleoperation] repository

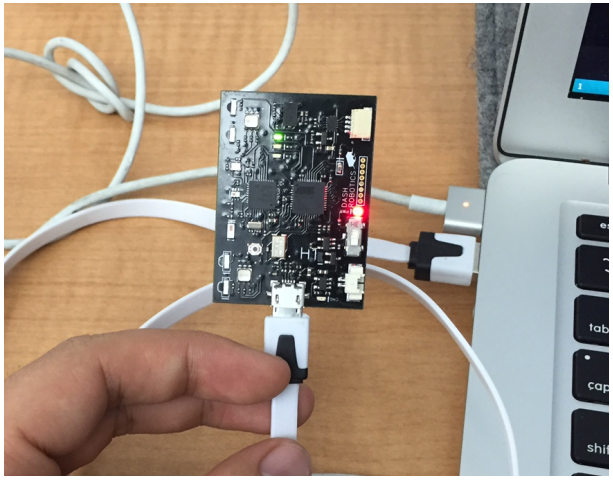


Only Dashboard from Dash Robotics Supported

2. Connect Battery and Motors

Motors: (fill find model numbers later)
Battery: (fill find connector labels later)
http://www.thunderpowerrc.com/Products/Ultra-Micro

Setup Microcontroller: Using Arduino Software with the Dashboard



Connect Dashboard to USB on Computer, turn on the Dashboard

1)

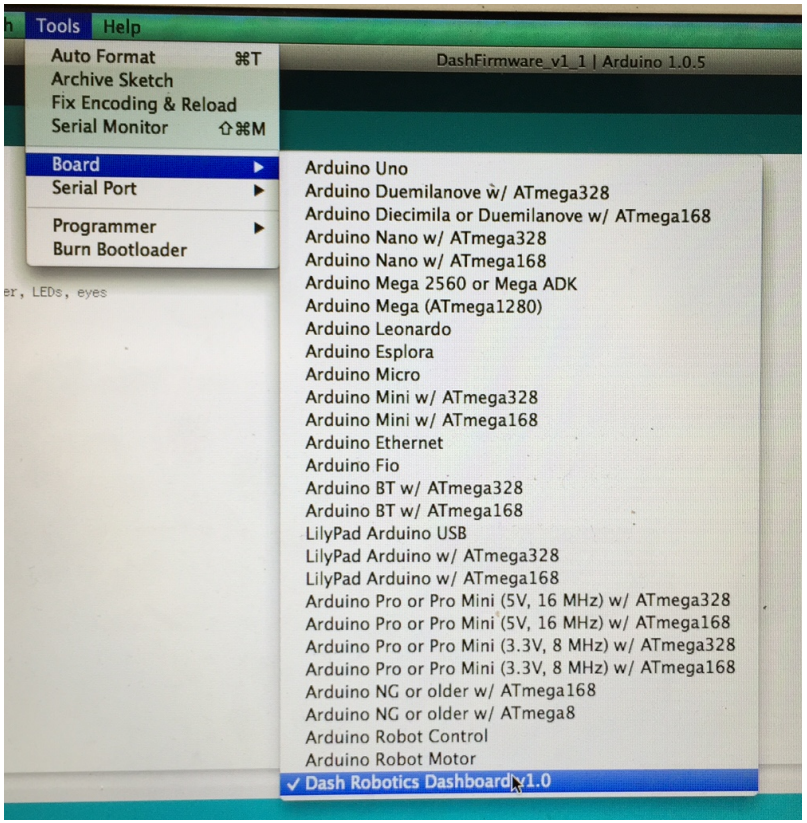
Download and Install Arduino Software (Arduino IDE)
http://arduino.cc/en/main/software

Open DashFirmware_v1_1.ino provided in
https://github.com/themichaeltsang/DashboardTeleoperation

2)

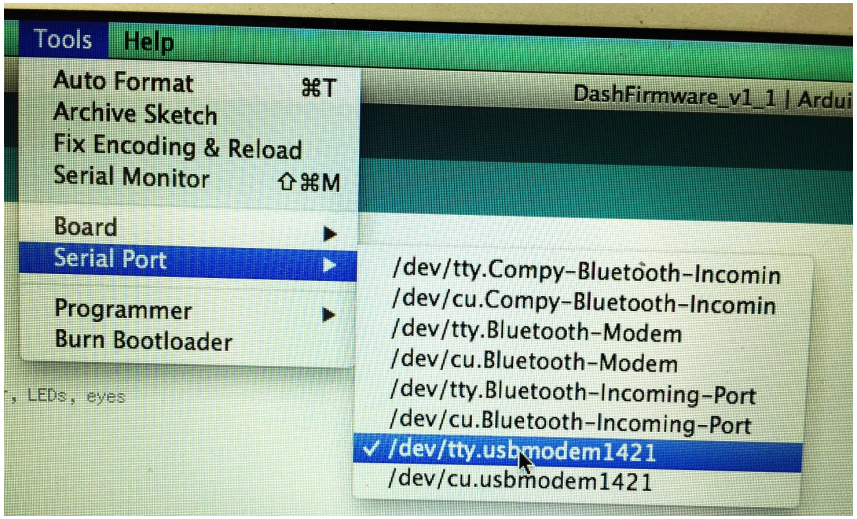
Note that the Dashboard runs custom Arduino UNO software, so it behaves like an arduino

Setup Microcontroller: Using Arduino Software with the Dashboard



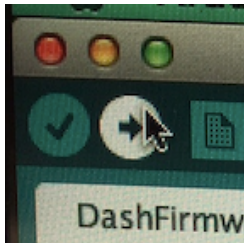
Select Dashboard from Tools > Board

3)



Select tty.usbmodemxxxx from Tools > Serial Port

4)



5) Press Upload

Note that the Dashboard runs custom Arduino UNO software, so it behaves like an arduino

Setup Computer

1. Use Mac Computer with OSX
   Tested with 2009 Macbook and newer
   Tested with OSX 10.9.2 and newer
2. Attach Bluetooth 4.0 Dongle if Computer doesn't have it.
   Set the dongle as default bluetooth by following these instructions:
   http://www.geekguides.co.uk/414/how-to-select-a-bluetooth-adapter-in-os-x/
3. Install Node js
   Click Install at: http://nodejs.org/
4. Download code repository at
   https://github.com/themichaeltsang/DashboardTeleoperation

Setup Computer: Text Editor

- Recommendation:
  - Use the text editor, sublime text : http://www.sublimetext.com/ (shown in slide 18)

    - Sublime text is easy on the eyes and nicely highlights keywords in many different programming languages
    - It also supports multi-window view, and multi-character selection.
    - My favorite commands are
      - "cmd ?" macro to comment out (deactivate) a selection of text
      - "Find All" after typing the "cmd f" macro, so you can, in one step, change a variable name appearing throughout an entire document to a new name
    - I recommend adding Sublime linter to auto-find syntax errors for you
      - http://youtu.be/uP2pkvHh2pY?list=PLLnpHn493BHEYF4EX3sAhVG2rTqCvLnsP

Setup Computer: Terminal Basics

- Terminal is a native program on Mac OSX
- Open Terminal
- Type ls
  - Will show you the contents of the folder you are currently in
- Type cd <folder name>
  - Enters a folder in the current directory
- Type cd ..
  - Moves you to folder enclosing the one you are currently in

- ls and cd are critical to navigating to the Teleoperation program directory, so you can run the program yourself

More guidance:
http://mac.appstorm.net/how-to/utilities-how-to/how-to-use-terminal-the-basics/

Sublime text and Terminal Workflow

Usage

Usage

1. Set Parameters in PARAMS.js in root directory
2. Run “node main.js” in the same directory using Terminal
3. Press user-defined keys to trigger robot behaviors. User-defined keys are set in PARAMS.js

If you need more guidance, consult:

https://github.com/themichaeltsang/DashboardTeleoperation

Usage: 1) Set Parameters

PARAMS.js

```
11
12  //EDIT PARAMETERS HERE
13      //TELEMETRY
14          define( 'Save_All_Sensor_Data', true ); //Output to CSV file
15          define( 'Print_Live_Sensor_Data_to_Terminal', false );
16          define( 'Rate_of_Sensor_Data_Arrival', 10 ); //(<= 100Hz)
17
18      //DC MOTORS
19          define( 'Forward_Speed', 80 ); //1-100
20          //define(    'Backward_Speed', 50 );
21          define( 'R_Turn_Angular_Velocity', 70 ); //1-100
22          define( 'L_Turn_Angular_Velocity', 70 );
23          define( 'Turn_Duration', 300 ); //(ms)
24
25      //SERVOS
26          define( 'Servo1_Port', MOSI );
27          define( 'Servo2_Port', SDA );
28          define( 'Servo3_Port', MISO );
29
30          define( 'Servo1_Initial_Position', 0 ); //0-180 degrees
31          define( 'Servo1_Final_Position', 180 ); //0-180 degrees
32          define( 'Servo1_Speed',  40 ); //0=no speed, 1-255 slower to faster
33          define( 'Servo2_Intitial_Position', 0 ); //0-180 degrees
34          define( 'Servo2_Final_Position',  180 ); //0-180 degrees
35          define( 'Servo2_Speed',  40 ); //0=no speed, 1-255 slower to faster
36          define( 'Servo3_Initial_Position', 0 ); //0-180 degrees
37          define( 'Servo3_Final_Position', 180 ); //0-180 degrees
38          define( 'Servo3_speed',  40 );
39
40      //KEYPRESS DEFINITIONS - trigger "on-the-fly" control
41          define( 'Start_Automation1_KEY', 'p' );
42          define( 'Start_Automation2_KEY', 'o' );
43          define( 'Start_Automation3_KEY', 'i' );
44
45          define( 'Run_Forward_KEY', 'w' );
46          //define(    'Run_Backward_KEY', 's' );
47          define( 'Turn_R_KEY', 'd' );
48          define( 'Turn_L_KEY', 'a' );
49
50          define( 'Set_Servo1_Init_Position_KEY', 'z' );
```

Default parameters shown

Usage: 2 & 3) Run in Terminal

```
Michaels-MacBook-Air-7:DashboardTelemetry Michael$ node main.js
SEARCHING FOR DASH
DASH DISCOVERED
        Initializing gyro
        Saving sensor data, appended to sensorlog.csv
RUNNING...

Press keys defined in PARAMS.js:
        Start Automode1: p
        Start Automode2: o
        Start Automode3: i
        Run Forward: w
        Turn Right: d
        Turn Left: a
        Set Servo1 initial: z
        Set Servo1 final: x
        Set Servo2 initial: c
        Set Servo2 final: v
        Set Servo3 initial: b
        Set Servo3 final: n
```

Run “node main.js”, press keys according to definitions (definitions will be shown in Terminal for your convenience)

Usage Notes

- Every keypress command triggers a response from the Dashboard near instantaneously.
- It's very fluid to control the Dashboard controller remotely
- Note that signals are being sent from the controller constantly, so sending signals back to the controller in a data stream at the same time may interrupt the bluetooth communication process

Telemetry

Telemetry Parameters

In PARAMS.js

```
//TELEMETRY
    define( 'Save_All_Sensor_Data', true ); //Output to CSV file
    define( 'Print_Live_Sensor_Data_to_Terminal', false );
    define( 'Rate_of_Sensor_Data_Arrival', 10 ); //(<= 100Hz)
```

Good for observing live changes
In sensor readings

This parameter enables or disables sensor data recording on the computer.
If true, data will be appended to sensorlog.csv. (see next slide)

If false, data will not be appended.

If you would like a new file to be made, delete sensorlog.csv and set to true, and a new sensorlog.csv will be created with new sensor data

This parameter sets the rate at which data will be emitted from the dashboard. Because some data processing is required to interpret the data from the computer, the computer does not read the data fast enough at high rates.

10 Hz is safe. You may try pushing the rate to 100 Hz

Telemetry File: sensorlog.csv

Sensor Data: Sun Dec 07 2014 12:37:49 GMT-0800 (PST)							
Expected Time (ms)	Actual Time (ms)	Yaw Rate (deg/sec)	Ambient Light	L Proximity	R Proximity	L Motor(%)	R Motor (%)
0	133	4	732	291	374	0	0
100	201	4	727	294	377	0	0
200	336	4	717	299	383	0	0
300	403	4	708	304	388	0	0
400	538	4	705	304	389	0	0
500	606	4	707	303	388	0	0
600	741	4	705	304	389	0	0
700	808	4	708	304	389	0	0
800	943	4	711	305	387	0	0
900	1011	4	708	307	389	0	0
1000	1146	4	706	309	390	0	0
1100	1213	4	706	308	390	0	0
1200	1348	4	705	309	390	0	0
1300	1416	4	707	308	390	0	0

At 10 Hz, Expected times of data arrival

Actual times of data arrival

Yaw Rate, Changes rapidly if you jerk the controller

Increases with ambient light intensity

The closer object are to the sensors, the smaller the values

The effort of each dc motor

Sensor Read Functions

Read Yaw Rate

```
readGyroDeg(function(yaw_rate){
    //YOUR CODE HERE that depends on yaw_rate
});
```

Read Ambient Light Intensity

```
 readAmbLight(function(amb_light){
    //YOUR CODE HERE that depends on amb_light
});
```

Read L and R Proximity

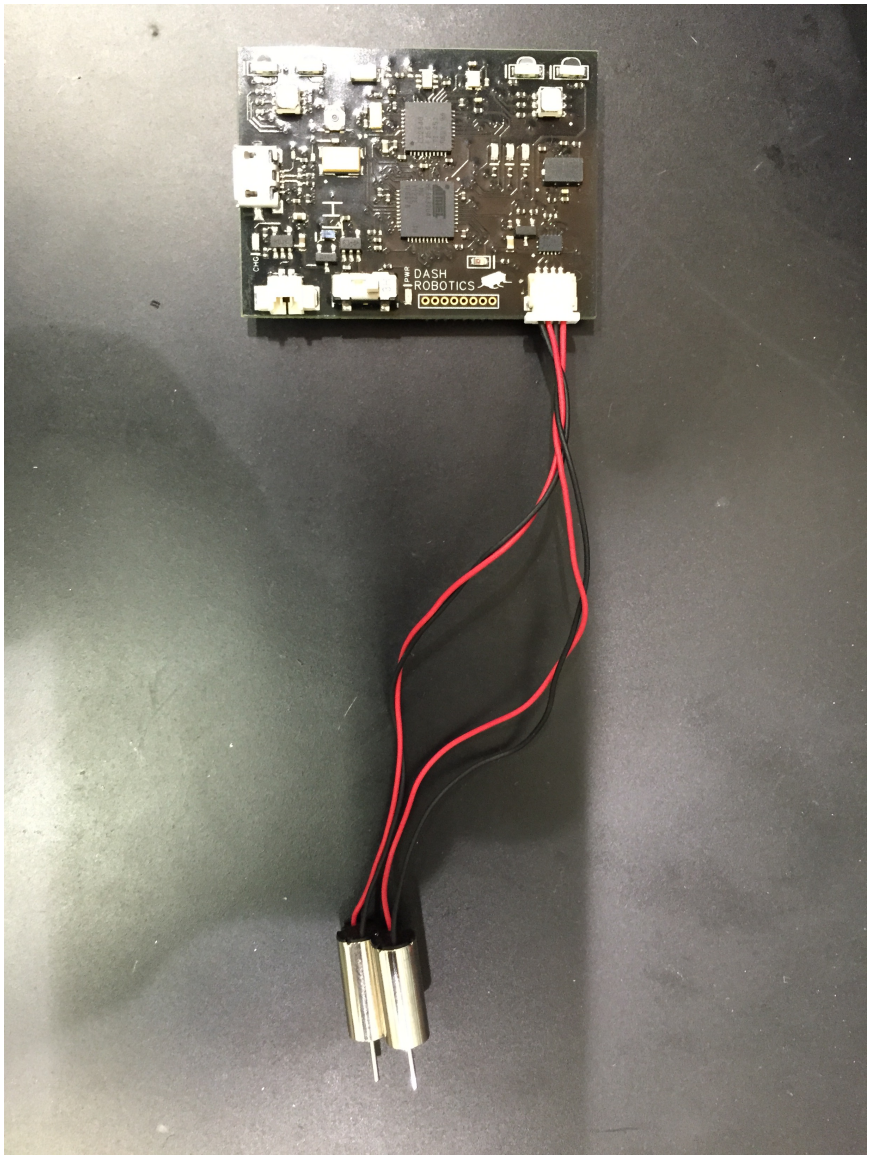
```
readLeftProximity(function(l_IR)){
    //YOUR CODE HERE that depends on l_IR
});

readRightProximity(function(r_IR)){
    //YOUR CODE HERE that depends on r_IR
});
```

These functions, to be used in main.js, are valid for automatically controlled events, triggered by:

Start_Automode1_KEY,
Start_Automode2_KEY,
or
Start_Automode3_KEY

Telecommand: DC & Servo Motors

DC Motor Connection



DC Motor Parameters

In PARAMS.js

```
//DC MOTORS
    define( 'Forward_Speed', 80 ); //1-100
    define( 'R_Turn_Angular_Velocity', 70 ); //1-100
    define( 'L_Turn_Angular_Velocity', 70 );
    define( 'Turn_Duration', 300 ); //(ms)
```

These parameters are only valid for manually controlled events, triggered by:

Run_Forward_KEY, Turn_R_KEY, or Turn_L_KEY

DC Motor Control Function

- Drive the Motors for a specified time

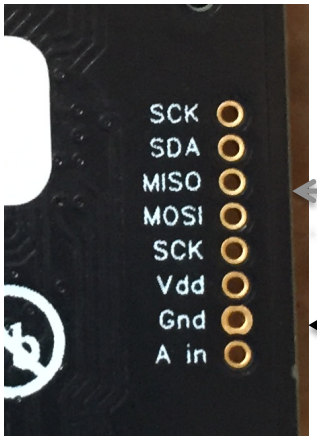
motorDrive (*left_motor_speed*, *right_motor_speed*, *duration_of_run*)

motor_speed acceptable range of values: 0 no speed, 1-100 slower to faster

These functions, to be used in main.js, are valid for automatically controlled events, triggered by:

Start_Automode1_KEY, Start_Automode2_KEY,
or Start_Automode3_KEY

Servo Motor Connection



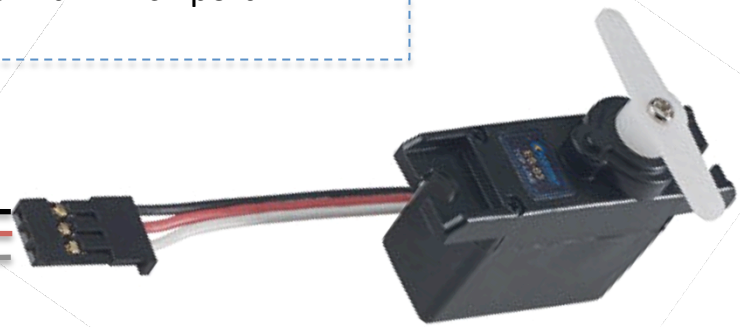
Dashboard backside

SDA, MISO, or MOSI

Gnd

```
define( 'Servo1_Port', MOSI );
define( 'Servo2_Port', SDA );
define( 'Servo3_Port', MISO );
```

In PARAMS.js, define which servo goes with which port



-

+

Battery

Check Servo spec sheet for its operating voltage range to pick a battery

If you'd like to add more servos (up to 3), you can use the same battery

Servo Motor Parameters

In PARAMS.js

```
define( 'Servo1_Initial_Position', 0 ); //0-180 degrees
define( 'Servo1_Final_Position', 180 ); //0-180 degrees
define( 'Servo1_Speed',  40 ); //0=no speed, 1-255 slower to faster
define( 'Servo2_Intitial_Position', 0 ); //0-180 degrees
define( 'Servo2_Final_Position',  180 ); //0-180 degrees
define( 'Servo2_Speed',  40 ); //0=no speed, 1-255 slower to faster
define( 'Servo3_Initial_Position', 0 ); //0-180 degrees
define( 'Servo3_Final_Position', 180 ); //0-180 degrees
define( 'Servo3_speed',  40 );
```

These parameters are only valid for manually controlled events, triggered by:

Set_Servo1_Init_Position_KEY, Set_Servo1_Final_Position_KEY,
Set_Servo2_Init_Position_KEY, Set_Servo2_Final_Position_KEY,
Set_Servo3_Init_Position_KEY, or Set_Servo3_Final_Position_KEY

Servo Motor Control Functions

servo1_control(*position*, *speed*)
servo2_control(*position*, *speed*)
servo2_control(*position*, *speed*)

position acceptable range of values: 0-180 degrees
speed acceptable range of values: 0-255, recommend 50. Passed around 70, servo may not move to desired position

These functions, to be used in main.js, are valid for automatically controlled events, triggered by:

Start_Automode1_KEY, Start_Automode2_KEY,
or Start_Automode3_KEY

Customization: User-defined Robot Behaviors

Program Customization: Basics

On the top of the main.js file:

```
function automode_sketch1() {
    servo1_control(180, 50); //t_0
    setTimeout(function(){
        motorDrive(100,100,800);
        }, 100 //t_1 (ms)
    );
    setTimeout(function(){
        motorDrive(100, 50, 8000);
        }, 1000 //t_2
    );
    setTimeout(function(){
        servo1_control(0, 50);
        servo2_control(0, 50);
        }, 4000 //t_3
    );
}
```

Start moving servo at t = 0

setTimeout: Delay the block of code until time: t = 100ms

At t = 1s, drive left motor at full capacity and right motor at half capacity for 8s

At t = 4s, actuate both servos, moving both of them to 0 degrees position at 50 speed

Program Customization: For loop

The “For” loop allows you to loop repeatable behavior:

What this code does is move servos by increments of 30 degrees in 1.5s intervals until the servos hit the position of 180 degrees.

All the while, the dc motors are moving at full speed uninterrupted.

```
function automode_sketch2() {
    for(var i = 0; i <= 180; i += 30){
        (function(i) {
            setTimeout(function() {
                    servo1_control(i,50);
                    servo2_control(i,50);
                    motorDrive(100,100,1500);
                }, 50*i
            );
        })(i);
    }
}
```

Run this block of code at times dependent of the iteration value i:

This allows you to loop repeatable behavior

Program Customization: Incorporating Sensor Readings

In this code the dc motors continuously run when the ambient light is dark, and they stop running in bright ambient lighting. The dc motors will run again whenever it is dark again. In this code, sensor data is read at 1 Hz.

```
function automode_sketch3() {
    for(var i = 0; i <= 10000; i += 10){
        (function(i) {
            setTimeout(function() {
                    readAmbLight(function(raw) {
                        console.log(raw);
                        if (raw<70){
                            motorDrive(100,100,1000);
                        }
                        else{
                            motorDrive(0,0,1000);
                        }
                    });
                }, 100*i
            );
        })(i);
    }
}
```

raw is the ambient light sensor reading, which is used in the next two lines of code

In-Depth: Information Transfer over Bluetooth

Bluetooth Low Energy

- Bluetooth Low Energy (BLE) was the key development in Bluetooth 4.0, and it is the wireless communication link between the dashboard and remote controllers.

Communicating with Bluetooth Low Energy

- The dashboard sends and receives “serial packets” to transfer data wirelessly to and from an external controller.

- These packets are 14 bytes long each.

- (each byte is 8 bits – basically you get 256 [0-255] unique numbers encoded in each byte)

Sending Packets in the Code - Example

Sending out serial from the computer (in main.js)

```
function servo1_control(degree, speed, callback){
    global.characteristic_write.write(new Buffer([9,degree,speed,1,0,0,0,0,0,0,0,0,0,0]), true, callback);
}
function servo2_control(degree, speed, callback){
    global.characteristic_write.write(new Buffer([9,degree,speed,2,0,0,0,0,0,0,0,0,0,0]), true, callback);
}
function servo3_control(degree, speed, callback){
    global.characteristic_write.write(new Buffer([9,degree,speed,3,0,0,0,0,0,0,0,0,0,0]), true, callback);
}
```

14 bytes:
Each entry is a byte ranging from numbers: 0 to 255

Here, I encode servo degree, speed, and servo ID in independent bytes

Receiving Packets in the Code - Example

Receiving those serial packets on the microcontroller to control the servos (in DashBot.cpp)

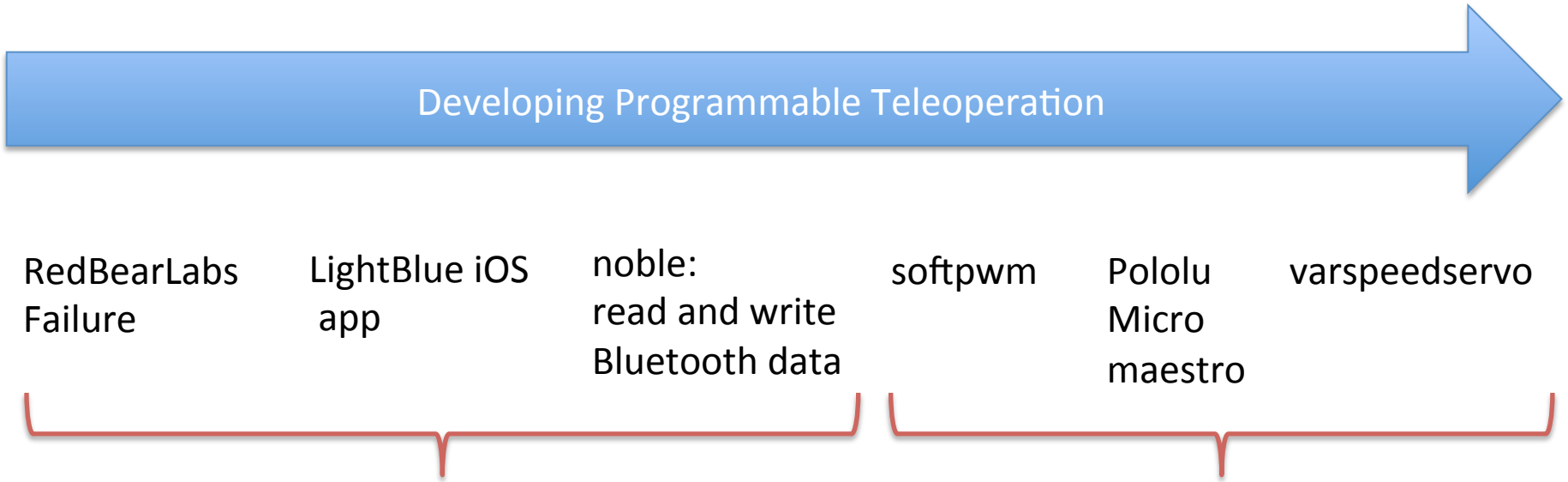
```
switch (receivedRadioPacket[0]){
  case 1: ...
  case 2: ...
  case 3: ...
  case 4: ...
  case 5: ...
  case 6: ...
  case 7: ...
  case 8: ...
  case 9:
      switch (receivedRadioPacket[3]){
        case 1:
          servo1.write(receivedRadioPacket[1], receivedRadioPacket[2], false);
          break;
        case 2:
          servo2.write(receivedRadioPacket[1], receivedRadioPacket[2], false);
          break;
        case 3:
          servo3.write(receivedRadioPacket[1], receivedRadioPacket[2], false);
          break;
        default:
          setEyeColor(100,0,100); //purple
          clearRadioPacket();
      }
    break;
```

Servo control methods, from the varspeedservo library

Those 14 bytes are received in the array, “receivedRadioPacket”. I take the encoded servo degree, speed, and servo ID, and write them to servo control methods

Previous Attempts and Failures

Research Path



Significance of Each Step: Getting Bluetooth Connection

- RedBearLabs Dongles:
  - Situation: Wanted to use the “BLE Mini” Bluetooth dongle to communicate with the Dashboard.
  - Advantage: works with PC
  - Problem: The dongle’s documentation was missing the BLE “write without response” command.
- LightBlue iOS App:
  - Situation: Needed more information on the bluetooth communication with the Dashboard
  - Solution: The LightBlueiOS App discovers the Dashboard, and you can send “write with response” commands to turn the Dashboard lights on. Further, you can read telemetry data with the “Notfy” command
- Noble BLE Central Interface:
  - Situation: Needed to be able to send “write without response” commands and send a “notify” command from the computer
  - Solution: the Noble BLE central interface by Sandeep Mistry works for Mac and Linux
  - Setback: No PC compatibility
  - Currently in use

Significance of Each Step: Controlling Servos from Dashboard Ports

- softpwm (google it):
  - Works to control servos directly for ports
  - Disadvantages: 1) Disables one direction of running control in a DC motor. 2) Servos jitters a lot. 3) can only set the servo to ~6 positions
- Pulolu Micro Maestro:
  - Situation: Wanted to find a way to avoid all the disadvantages of softpwm
  - Problem: Haven't been able to get it work via a softserial connection between the dashboard and pulolu micro maestro yet. Haven't even been able to get communication working between dasboard and Arduino uno, although people on the internet have.
  - Disadvantage: the Micro Maestro adds extra weight as it is a separate controller chip
- varspeedservo:
  - Same as softpwm without disadvantages 2) and 3).
  - Main Advantage: 180 degrees of positions able to be set, easily controllable servo speed
  - Main Disadvantage: Still disables one direction of running control in a DC motor.
  - Currently in use

Next Steps

- Enable backward movement when also controlling servos
- Add SPI interface for more sensor options
- Increase the frequency of sensor access in automode for more responsive feedback
  - Current problem may be “settimeout” method or conflicting Bluetooth signals
- Add intelligent control algorithms