gRover Robotics Platform v1.0

888888888ba 888888888ba
88 "8b 88 "8b
88 ,8P 88 ,8P
,adPPYb,d8 88aaaaaa8P' ,adPPYba, 8b d8 ,adPPYba, 88aaaaaa8P'
a8" `Y88 88''''88' a8" 88a `8b d8' a8P_____88 88''''88'
8b 88 88 `8b 8b d8 `8b d8' 8PP'''''''''' 88 `8b
"8a, ,d88 88 `8b "8a, ,a8" `8b,d8' "8b, ,aa 88 `8b
"YbbdP"Y8 88 `8b "YbbdP"' "8" "Ybbd8"' 88 `8b
aa, ,88 "Y8bbdP"
gRoveR, a SeeedStudio Grove Toy Kit Contest Entry, 25-Aug-2011

Matthew Lange, http://erroraccessdenied.com

-------------------------------------------
Matthew Lange, http://erroraccessdenied.com
-------------------------------------------
Purpose:
To re-purpose a remote control car that I obtained from the thrift store for $2 into an Arduino-controlled robot using Seeed Studio's “Grove” platform.

Description:
I purchased an RC car from the thrift store for $2. It had no remote control to go along with it, but it did have a 4.8V Ni-Cad battery pack, a chassis, a steering motor, and a drive motor. While the 4.8V Ni-Cad battery does not have enough voltage to power the I2C Motor Driver (it may be useful in another project), the motors and chassis provide a great base for a robot.

With the Grove platform, I endeavored to turn it into an Arduino-based rover with the following modes:
- **Autonomous Mode:** Drive the rover around autonomously using the on-board sensors for guidance
- **Remote Control:** Drive the rover around with commands sent over the serial port
- **Cat Spooker:** Lie in wait until a cat approaches, then make some noise and start Autonomous mode
- **Cat Taunter:** Make a noise every 15 seconds while there isn't a cat in the area.
- **Motor Test:** Run a test of the motors connected to the I2C Motor Driver Twig
- **Sensor View:** Get a live view of sensor data

Parts List: This project uses the following parts:

<table>
<thead>
<tr>
<th>Qty</th>
<th>Part Name</th>
<th>SKU#</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R/C Truck</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Arduino (or Arduino Clone)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Battery Box (Output: 8-15V)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Stem – Base Shield</td>
<td>SLD12148P</td>
</tr>
<tr>
<td>1</td>
<td>Twig – OLED Display 128x64</td>
<td>OLE35046P</td>
</tr>
<tr>
<td>1</td>
<td>Twig – I2C Hub</td>
<td>ACC53133P</td>
</tr>
<tr>
<td>1</td>
<td>Twig – I2C Motor Driver</td>
<td>ROB72212P</td>
</tr>
<tr>
<td>1</td>
<td>Twig – I2C Touch Sensor</td>
<td>SEN51153P</td>
</tr>
<tr>
<td>1</td>
<td>Twig – Sound Recorder</td>
<td>SEN71254P</td>
</tr>
<tr>
<td>1</td>
<td>Twig – Buzzer</td>
<td>COM22458P</td>
</tr>
<tr>
<td>1</td>
<td>Twig – PIR Motion Sensor</td>
<td>SEN32357P</td>
</tr>
<tr>
<td>1</td>
<td>Twig – 80cm Infrared Proximity Sensor</td>
<td>SEN39046P</td>
</tr>
<tr>
<td>1</td>
<td>Twig – 3-Axis Accelerometer</td>
<td>SEN21853P</td>
</tr>
<tr>
<td>2</td>
<td>Twig – Chainable RGB LED</td>
<td>COM53140P</td>
</tr>
</tbody>
</table>
gRover Robotics Platform v1.0

1  Twig – Sound Sensor  SEN12945P
1  Twig – Vibrator  ROB51043P
3  Grove – Universal 4-Pin Cable (5 ea)  ACC113170

Part Descriptions:

The Twigs perform the following tasks:

- The 3-axis Accelerometer Twig helps track the rover's movements, and can be used to tell if it is standing still (despite motors turning).
- The OLED Display 128x64 Twig and I2C Touch Sensor Twig are used as a User Interface, mounted on the top of the robot (or under a clear polyurethane shell for weatherproofing). The Vibrator Twig is used as tactile feedback for the I2C Touch Sensor Twig's buttons, (also known as “feelers”) which aren't physical buttons but actually touch sensors.
- The Buzzer Twig is used as audible feedback in addition to the Vibrator Twig's tactile feedback, and can be used for sound effects if the Sound Recorder Twig isn't used.
- The PIR Motion Sensor Twig is set to a fairly short range, and is used to either detect obstacles, or (in some cases) wait for obstacles (like cats!) to approach it.
- The 80cm IR Proximity Sensor Twig is used to detect obstacles ahead of the rover.
- The Sound Recorder Twig has various sounds recorded, such as a monster-truck engine and a car horn.
- The Chainable RGB LED Twigs act as pseudo-Police flashing lights (with two RGB LEDs, flashing in a red/blue pattern)
- The Sound Sensor Twig is used to control the robot by clapping.
- The I2C Motor Driver Twig is used to control the drive and steering motors in the RC car.
Assembly Instructions:

### Stem Base Shield

<table>
<thead>
<tr>
<th>Digital TWIG Ports¹</th>
<th>Analog TWIG Ports²</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <strong>(Do not use – Serial)</strong></td>
<td>1. Sound Sensor Twig</td>
</tr>
<tr>
<td>2</td>
<td>2. 80cm IR Proximity Sensor Twig</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4. <strong>(Do not use – I2C)</strong></td>
</tr>
<tr>
<td>5</td>
<td>5. <strong>(Do not use – I2C)</strong></td>
</tr>
<tr>
<td>6</td>
<td><strong>I2C TWIG Ports</strong></td>
</tr>
<tr>
<td>7</td>
<td>7. Any I2C Hub Twig</td>
</tr>
<tr>
<td>8</td>
<td>8. Any I2C Motor Driver Twig</td>
</tr>
<tr>
<td>9</td>
<td><strong>UART TWIG Port³</strong></td>
</tr>
<tr>
<td>10</td>
<td>10. <strong>(Future: Bluetooth Twig?)</strong></td>
</tr>
<tr>
<td>11</td>
<td><strong>SPI TWIG Port</strong></td>
</tr>
<tr>
<td>12</td>
<td>12. PIR Sensor Twig</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I2C Hub TWIG Ports⁴</th>
<th>I2C Motor Driver Ports⁵</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stem Base Shield I2C Port</td>
<td>Battery</td>
</tr>
<tr>
<td>I2C Accelerometer Twig</td>
<td>VS Battery V+ (6-15V)</td>
</tr>
<tr>
<td>I2C Touch Sensor Twig</td>
<td>GND Battery GND (0V)</td>
</tr>
<tr>
<td>I2C OLED Display 128x64</td>
<td>Motor 1</td>
</tr>
<tr>
<td></td>
<td>M1- Motor 1's GND</td>
</tr>
<tr>
<td></td>
<td>M1+ Motor 1's 5V</td>
</tr>
<tr>
<td></td>
<td>Motor 2</td>
</tr>
<tr>
<td></td>
<td>M2- Motor 1's GND</td>
</tr>
<tr>
<td></td>
<td>M2+ Motor 1's 5V</td>
</tr>
<tr>
<td></td>
<td>Jumper</td>
</tr>
<tr>
<td>Digital Pin 2</td>
<td>J4 Connect to power the Arduino from the motor driver's battery⁷</td>
</tr>
<tr>
<td>Digital Pin 3</td>
<td>I2C Touch Sensor</td>
</tr>
<tr>
<td>I2C Accelerometer</td>
<td>I2C Motor Driver</td>
</tr>
</tbody>
</table>

---

1. Digital TWIG Ports are as numbered in silkscreen above each white TWIG connector; each Digital TWIG port gives access to 2 Arduino pins, ie Port 1 gives access to Arduino Digi Pins 1 and 2; Port 10 gives access to Pins 10 and 11. Not all TWIGs use both pins from their port.
2. Analog Port 1 is closest to the Stem Base Shield RESET button and handles pins A0 and A1.
3. Adding a UART-capable TWIG here will cause the USB serial port on the Arduino to not function.
4. I2C Hub Twig's devices can be attached in any order
6. Attach the pins on the Arduino (shown on left) to the INT pins on the TWIGs (shown on the right) with a piece of wire
7. Don't connect the jumper J4 if the Arduino is powered by any other source. Connecting the jumper sends power over the I2C Bus to the Arduino and everything else connected to it.
gRover Robotics Platform v1.0

- Attach all TWIGs, wires, and motors as described above.
- Attach the PIR Sensor Twig to the robot's chassis so it is facing to the rear
- Attach the 80cm IR Proximity Sensor Twig to the robot's chassis so it is facing forward
- Mount the I2C Touch Sensor “Feelers” near the OLED Display for ease of use, in a diamond pattern with I2C Touch button 0 (“Up”) at the top, 1 (“Down”) at the bottom, 2 (“OK”) to the right, and 3 (“Cancel”) to the left.
- Affix the Buzzer Twig on the same surface as the I2C Touch Sensor “Feelers” so you can have noticeable tactile feedback

The assembly of this unit consisted mostly of fitting the modules wherever they would fit, within reason. This involved the cutting of plastic to make room for placing cabling where it was not originally intended to fit. I used black electrical tape for anything that was not a suitable task for the 3M SJ3000 8(Digikey# 3M10286-ND).

The I2C Hub, Sound Recorder, I2C Accelerometer and I2C Touch Sensor Hub are located inside the part of the plastic case where the original PCB was housed.

The Sound Recorder Twig's speaker and the Buzzer Twig are located under the front bumper of the rover, as there is a fair-sized cavity there that affords them some protection.

The OLED Display Twig, Touch Sensor Feelers and Buzzer Twig are located on a piece of corrugated cardboard at the rear of the rover. The cardboard is mounted over pegs where the R/C car's “shell” was screwed down. The pegs also act as a stabilizer for the Arduino (it fits nicely in the corner they form), and the cardboard helps lift the rear of the Arduino up to be level with a bump in the plastic.

The Chainable RGB LEDs are mounted on a piece of corrugated cardboard near the front of the rover, along with the Sound Sensor. I purposely kept the sound sensor away from the drive motor and drive wheels, as they would be the noisiest parts of the robot.

I put an 8xAA battery box where the 4.8V Battery Pack used to be. The batteries are secured into the battery box box using 3M SJ3000, and the battery box is secured to the rover with it as well. (Various loose cables can be grouped and stuck to it as well)

For additional construction tips and images, please see the annotated photos in my Flickr photoset at http://www.flickr.com/photos/matthewlange/sets/72157627614719221/with/6177085400/. I would have added the pictures here, though they wouldn't allow for enough detail and would have created a gigantic file.

---

8 I received a sample roll of 1” x 5 yards of SJ3000 Red. It is almost like double-sided tape, except that instead of adhesive, it uses Velcro-like hooks on one side and loops on the other, making it re-stickable.
How to Use:

- The I2C Touch Sensors are your primary inputs into the robot. The OLED display acts as your screen. I2C Touch button 0 is “Up”, 1 is “Down”, 2 is “OK”, 3 is “Cancel”.
  - I have also included a sheet to go over your sensors (with labels for your buttons) in an attached .xls file
- Press “Up” or “Down” to move the cursor to your desired option, and press “OK” to activate the function.
- To exit an option, press and hold “Cancel” until you feel the vibrator buzz. Depending on the function, it may be instantaneous.

Code: Compile and load the attached code to your Arduino.

Functions:

- **Initialization Functions**
  - void MMA7660Accel_init(void)
  - void i2cMotorDriver_init(void)
  - void i2cTouchSensor_init(void)
  - void rgbLed_init(void)
  - void I2cOledDisplay_init(void)
- **OLED Functions**
  - void setXY(unsigned char row, unsigned char col)
  - void sendChar(unsigned char ascii=0)
  - void sendStr(char* string)
  - void oled_clearDisplay(void)
- **Generic I2C Functions**
  - void sendI2cData(unsigned char i2cAddress, unsigned char commandMode, byte myData)
  - void sendI2cData(unsigned char i2cAddress, unsigned char commandMode, byte myData, byte myData2)
  - void sendI2cData(unsigned char i2cAddress, unsigned char commandMode, byte myData, byte myData2, byte myData3)
  - char requestI2cData(unsigned char i2cAddress, unsigned char commandMode, byte numBytes)
- **I2C Motor Driver Functions**
  - void setMotor(unsigned char motorName, unsigned char motorDirection, unsigned char motorSpeed)
  - void setMotors(unsigned char motorASpeed, unsigned char motorBSpeed)
  - void motorTest(unsigned char motorNum, unsigned char motorDirection, unsigned char motorSpeed)
- **Chainable RGB LED Functions**
  - void ClkProduce(void)
    - Clocks the RGB LED clock pin
gRover Robotics Platform v1.0

- void Send32Zero(void)
  - Sends 32 Zeros on the RGB LED data pin (to initialize or finalize communications)
- unsigned char TakeAntiCode(unsigned char dat)
  - Computes the checksum required by the RGB LEDs' IC
- void DatSend(unsigned long int dx)
  - Clock out the data in [dx] (32 bits) to the RGB LEDs
- void DataDealWithAndSend(unsigned char r, unsigned char g, unsigned char b)
  - Set the next available RGB LED to value [r,g,b] and send it.
- void rgbPoliceLights(void)
  - Flash the RGB LED(s) in a police-car-like pattern

  **I2C Touch Sensor Functions**
  - void buttonHandler(byte what)
    - Handles button presses from the I2C Touch Sensors
      - byte what Which button was pressed? (0 to 3)
  - char getButtonStatus(char buttonNum)
    - Returns the value of button [buttonNum] (1 or 0)
  - char handleTouch(void)
    - Gets called when a function notices that the flag has been set

  **Other Sensor Functions**
  - void readAccel(void)
    - Reads the acceleration from the accelerometer and save it to AccelX, AccelY, AccelZ
  - void vibratingMotor_buzz(int length)
    - Buzzes the vibrating motor for "length" milliseconds
  - void soundRecorder_playSound(int pin1, int pin2)
    - Asserts the value (0 or 1) of pin1 and pin2 on the two Sound Recorder Twig data pins

  **Menu System Functions:**
  - void showMainMenu(void)
    - Displays the Main Menu
  - void setArrow(int row)
    - Puts the ‘=>’ arrow on row [row] indicating the current selection
  - void createScreen(char** screenLines, char screenId, char* screenTitle, char menuUnderlineChar, char xOffset, char yOffset, char showCursor, int defaultRow)
    - Makes a screen; either a menu or a regular screen of text
      - char** screenLines An array of the row texts (MAX 6; ie. 0-5)
      - char screenId The ID number of the screen we're showing (so we can keep track of what to do when buttons are pressed)
      - char* screenTitle The title of the menu
      - char menuUnderlineChar The character we use to underline the title (typically 0x2d, "-")
      - char xOffset What position does text start in on the X-Axis? This
gRover Robotics Platform v1.0

... lets you leave room for the cursor (either "0" if you're not using a cursor, or "2" if you're using a 2-character cursor)

- **char yOffset**
  What position does text start in on the Y-Axis? This leaves room for the title.

- **char showCursor**
  Do we show the cursor? (ie, is this a menu row?)

- **int default_row**
  The default menu item to point at (0-5 with a title, 0-7 without)

- **void createScreen(char** screenLines, char screenId, char* screenTitle, char menuUnderlineChar)
  - Overload the real createScreen() with a shorter version for non-menu but titled screens with a custom underline character

- **void createScreen(char** screenLines, char screenId, char* screenTitle)
  - Overload the real createScreen() with a shorter version for non-menu but titled screens (with default 0x2d ("\"\") underline)

- **void createScreen(char** screenLines, char screenId)
  - Overload the real createScreen() with a shorter version for non-menu and non-titled screens

- **void menuSelect(char menuItem**
  - Perform the action associated with menu item #[menuItem]

- **Graphing Functions**
  - **void createGraph(char** title, int maxVal, int minVal)
    - Create a graph with the title [title] and a minimum and maximum Y value of [minVal] and [maxVal] respectively.
    - Not currently implemented.

  - **void plotValue(int x, int y**
    - Create a data point at (x,y)
    - Not currently implemented.

- **Robot Modes**
  - **void doAutonomous(void**
  - **void doCatSpooker(void**
  - **void doCatTaunter(void**
  - **void doRemoteControl(void**
  - **void doMotorTest(void**
  - **void doSensorView(void**

- **Interrupt Request Handlers**
  - **void onAcceleration(void**
  - **void onButtonPress(void**

- **Standard Functions**
  - **setup()**
  - **loop()**
gRover Robotics Platform v1.0

Repurposing:
The code was written with re-use and modification in mind: its modularity helps facilitate ease of programming and ease of control, while also allowing for easy upgrades and addition of code. The code is also well-commented, so anyone wishing to use bits and pieces will be able to figure out how and why it works.

Future Uses: With the right Twigs, users can (fairly easily) enhance the rover to add features such as:
- A “Compass” Twig for navigation
- A “Line Finder” Twig to make a line-following robot
- A “Light Sensor” Twig to make a light-following robot
- An “I2C Color Sensor” Twig to make a color-following robot (have it chase a ball, etc?)
- A “Serial Bluetooth” or “Serial RF Pro” Twig for remote control
- A “Temperature and Humidity Sensor Pro” to measure and plot temperature and humidity in a house or other area
- A “Gas Sensor” or “Geiger Counter” Twig to detect potentially hazardous environments
- A GPS mounted on a Protoshield Twig for (very rough) outdoor location detection

Future of this Project:
In code, I plan to implement:
- Graphing of individual sensor data on the OLED display
- RGB LED status indicators of how well the autonomous mode is handling its environment

With Protoshield Twigs, I plan to implement:
- GPS over Serial or I2C
- Various sensors I have samples of (ie Temperature, EEPROM, etc) over I2C
- A MiniSense 100 Piezo Sensor on Analog pins
- A Bluetooth Serial Module on Digital pins 0 and 1 for control from a phone or other device

With Seeed Studio Twigs, I plan to implement:
- An 80cm Proximity Sensor for better obstacle tracking than the PIR sensor is capable of
- A Line Finder Twig to have the robot follow a predetermined path for collecting sensor results

In hardware, I plan to implement:
- A small fan or stick-on heatsink(s) to keep the I2C Motor Driver and Voltage Regulator cool

Conclusion:
Seeed Studio's Grove kit makes it easy (and fun!) to create a rover bot. With the implementation I have put together, with sources cited in the .pde file, anyone can use some (or all) of these features to help create their own robots.

Errata:
1. If you attempt to use Serial.write() or Serial.read() (as of Arduino 0022) you may accidentally cause some very strange bugs (such as OLED glitches, I2C touch not working, or random

M. Lange http://erroraccessdenied.com
crashes and resets of the Arduino). This likely occurs due to the Wire and Serial libraries attempting to do things with the same chunks of silicon. To this extent, I have commented out most Serial.write() calls. Just keep this in mind when uncommenting them. At this time, it may be advantageous to use the NewSoftSerial Library to perform serial functions such as Bluetooth communications or debugging over USB.

2. Depending on your IDE and IC version, you may need to change the delay() values to have the code perform as expected.

3. I used chars a lot in the code to reduce the overall memory footprint vs. integers

4. The OLED Display is upside-down in the photos on flickr. The text on the screen displays in the opposite direction of the “OLED Display 128x64” text.

Sources:

- **Accelerometer:** Freescale MMA7660
  - WIKI: [http://seeedstudio.com/wiki/Twig_-_I2C_3-axis_Accelerometer](http://seeedstudio.com/wiki/Twig_-_I2C_3-axis_Accelerometer)

- **Buzzer:**

- **I2C Touch Sensor:** Freescale MPR121
  - Code Example: [http://www.prizepony.us/bin/arduino_mpr121.zip](http://www.prizepony.us/bin/arduino_mpr121.zip)

- **128x64 OLED Display:** Solomon SSD1308 (Controller) and Liyuan LY190-128064 (OLED)
  - WIKI: [http://seeedstudio.com/wiki/Twig_-_OLED_Display_128*64](http://seeedstudio.com/wiki/Twig_-_OLED_Display_128*64)

- **I2C Motor Driver:** STMicroelectronics L298 and Atmel ATmega8

- **Chainable RGB LED:** P9813

- **IR Proximity Sensor:** Sharp GP2Y0A21YK
  - WIKI: [http://seeedstudio.com/wiki/index.php?title=Twig_-_80cm_Infrared_Proximity_Sensor_v0.9](http://seeedstudio.com/wiki/index.php?title=Twig_-_80cm_Infrared_Proximity_Sensor_v0.9)

- **PIR Motion Sensor:** BluNet Intl. RE200B
  - WIKI: [http://seeedstudio.com/wiki/Twig_-_PIR_Motion_Sensor](http://seeedstudio.com/wiki/Twig_-_PIR_Motion_Sensor)

- **Vibrator:**

- **Sound Sensor:** LM386

- **Sound Recorder:** APlus Inc. APR9600
  - WIKI: [http://seeedstudio.com/wiki/Twig_-_Sound_Recorder_v0.92b](http://seeedstudio.com/wiki/Twig_-_Sound_Recorder_v0.92b)

**Disclaimer:** The usual disclaimers apply. I'm not responsible for damages caused by the use or misuse of this code or instructions. Use this at your own risk and take normal safety precautions.

M. Lange [http://erroraccessdenied.com](http://erroraccessdenied.com)