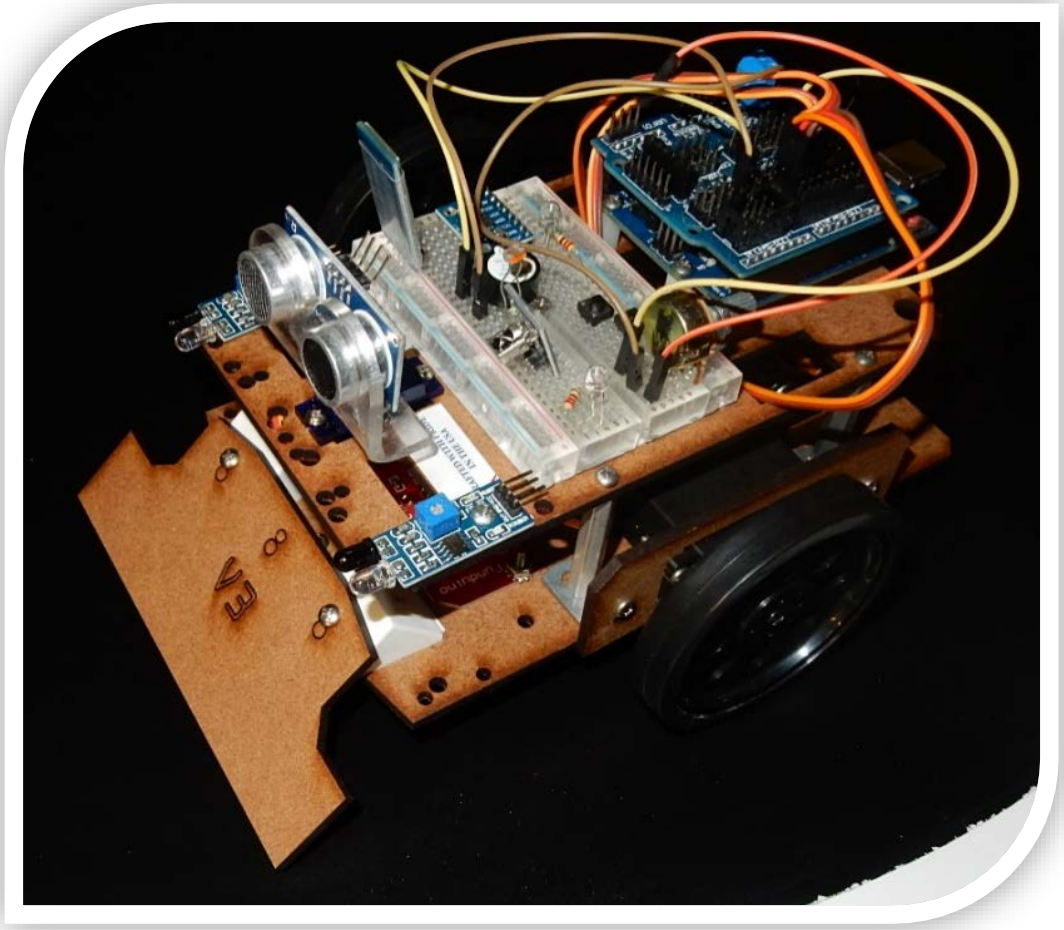
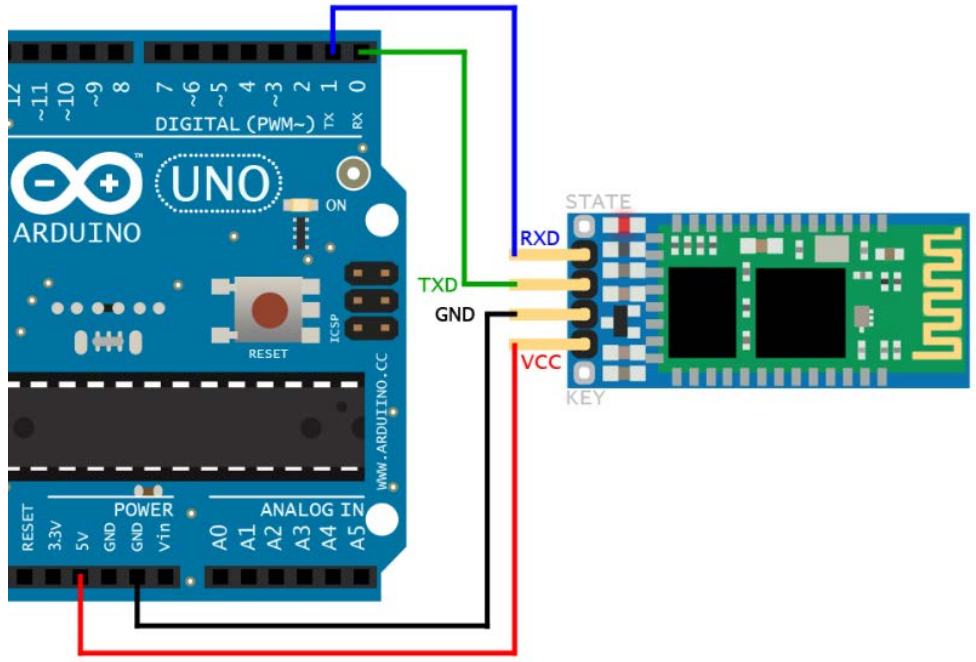
Experiment 7: Robotics++ V3 Robot BlueToothbot

ROBOTICS++

RESEARCH & EDUCATION

www.roboticsplusplus.com





Two different ways to control your robot via Bluetooth

1. Android phone – wire your robot, download apps from the Google Play Store or install an APK (app file) manually into your phone by downloading it and installing or create your own APK using the MIT application builder and Arduino code that receives serial commands from these apps to make your robot do things.
2. Computer – using a Bluetooth capable PC create an application using the Processing language and Arduino code to control your robot remotely. You can create buttons on the screen to make motors move or read sensors and turn lights or sound.

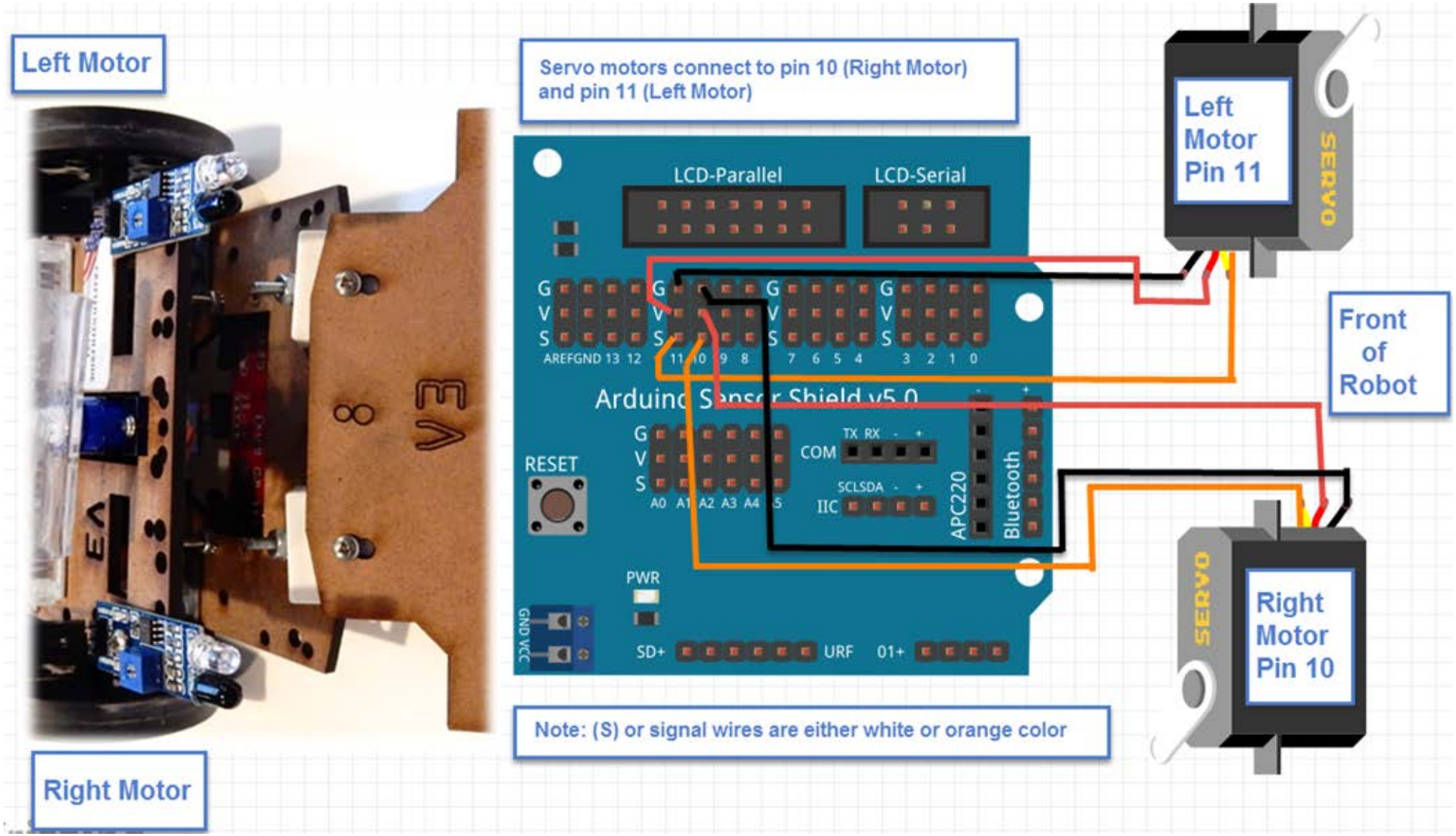
Important: before downloading the Arduino code to the robot you must unplug the plus or positive cable of the Bluetooth module otherwise you may not be able to upload the code.

Let's make sure our robot is wired correctly

Parts:
(2) continuous rotation servos

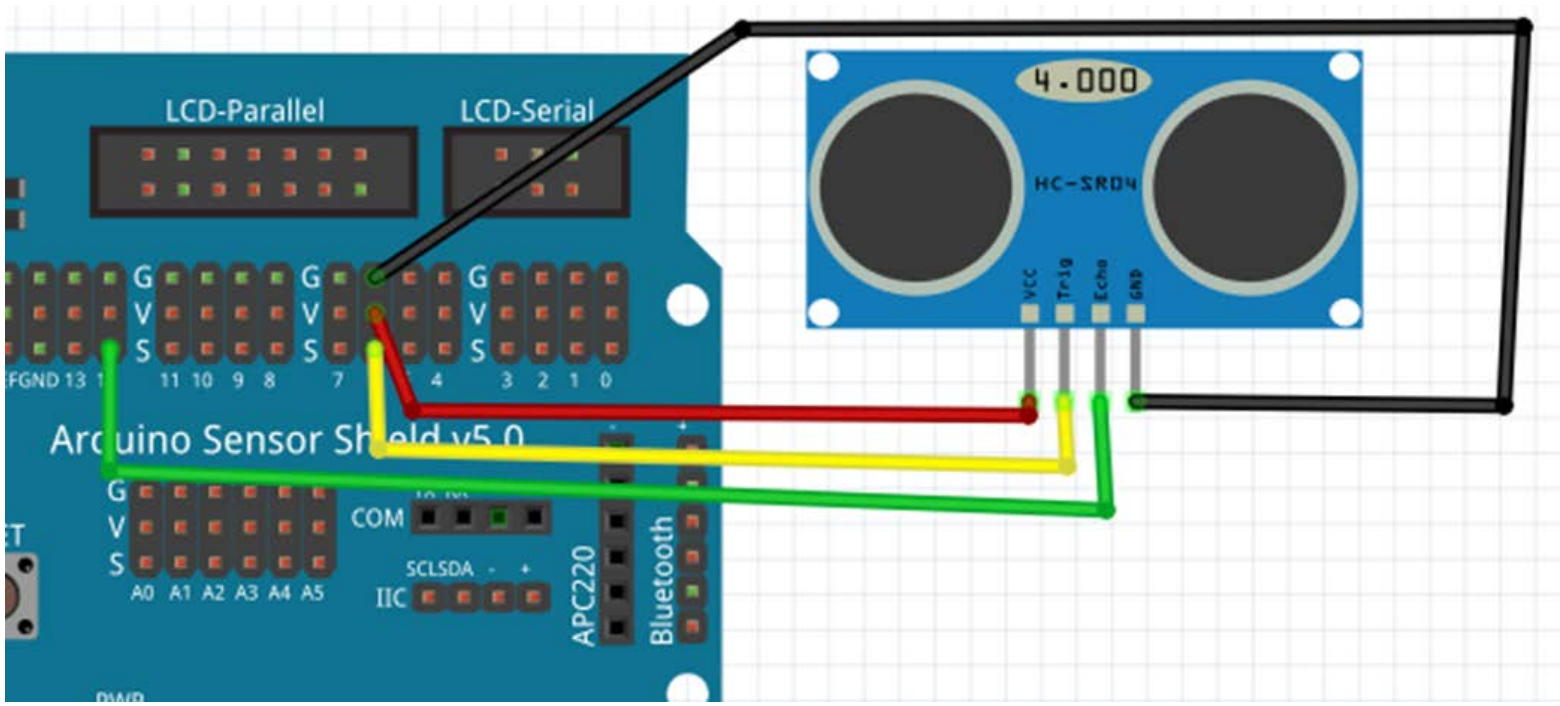
Servo motors wires connect directly on the sensor shield. Notice which one goes into pin 11 and pin 10.

Sumo ramp is not necessary



Installing the Sonar Sensor

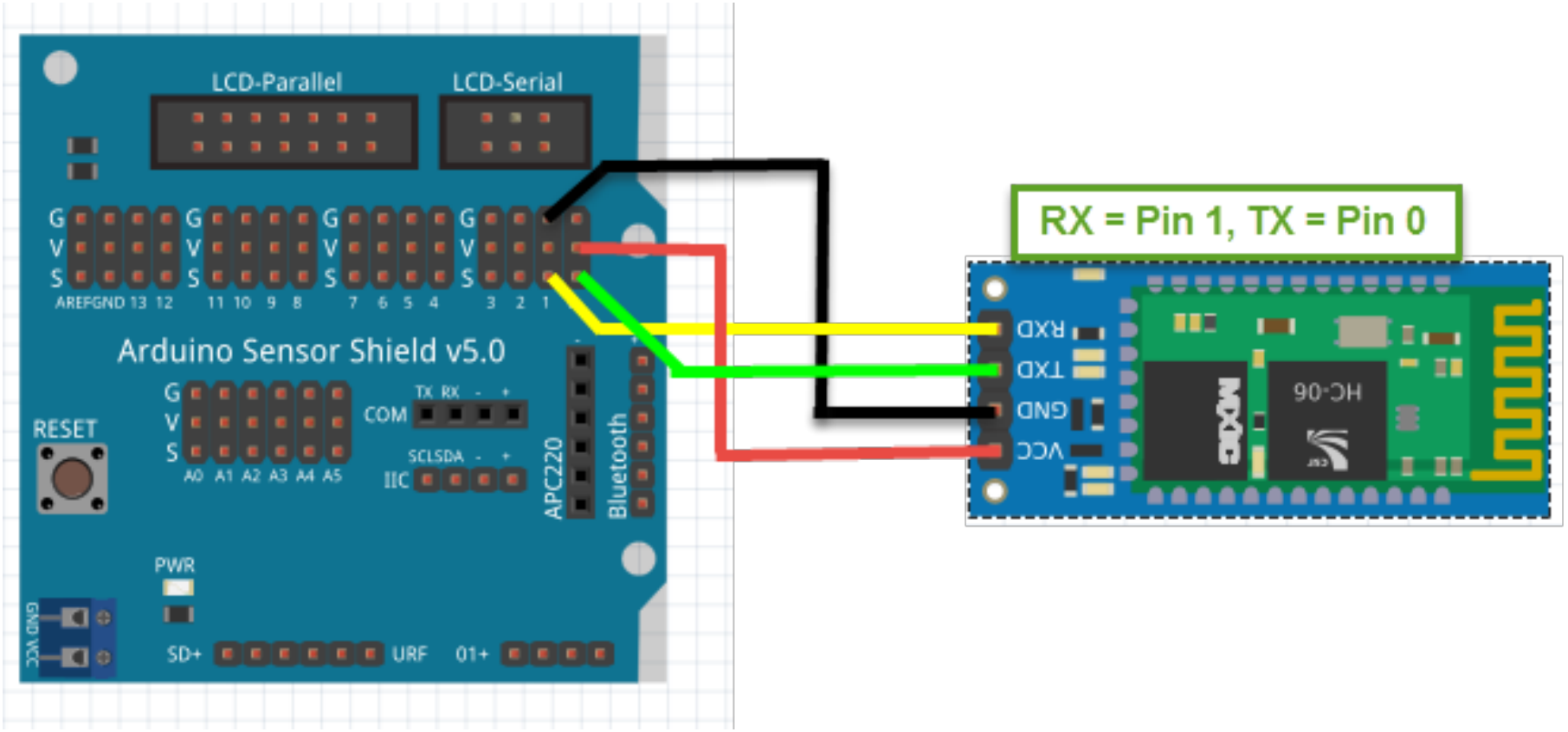
We are going to use the sonar sensor in this experiment to allow the robot to roam about on its own when not being controlled remotely



VCC =V, GND=G, TRIG=S(pin 6) and ECHO=S(pin12)

Installing the Bluetooth Module HC-5 or HC-06

Note: Pairing Password: 1234

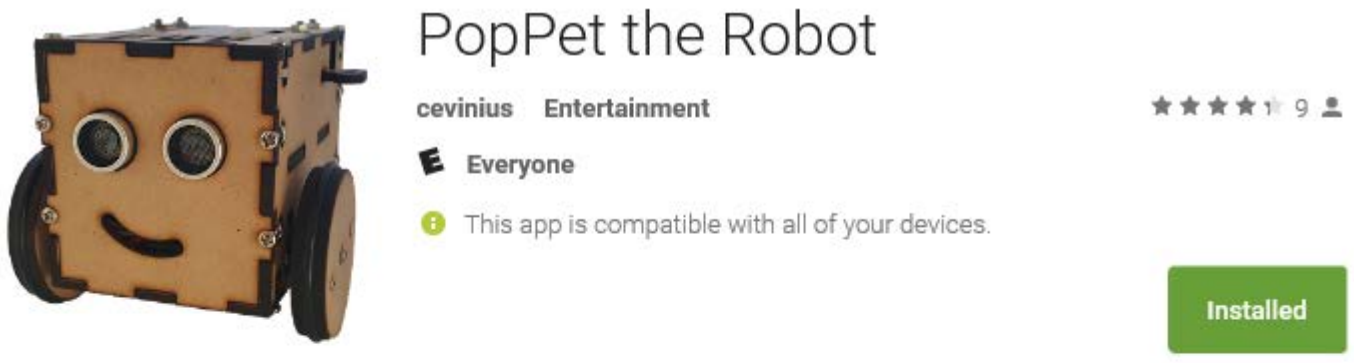


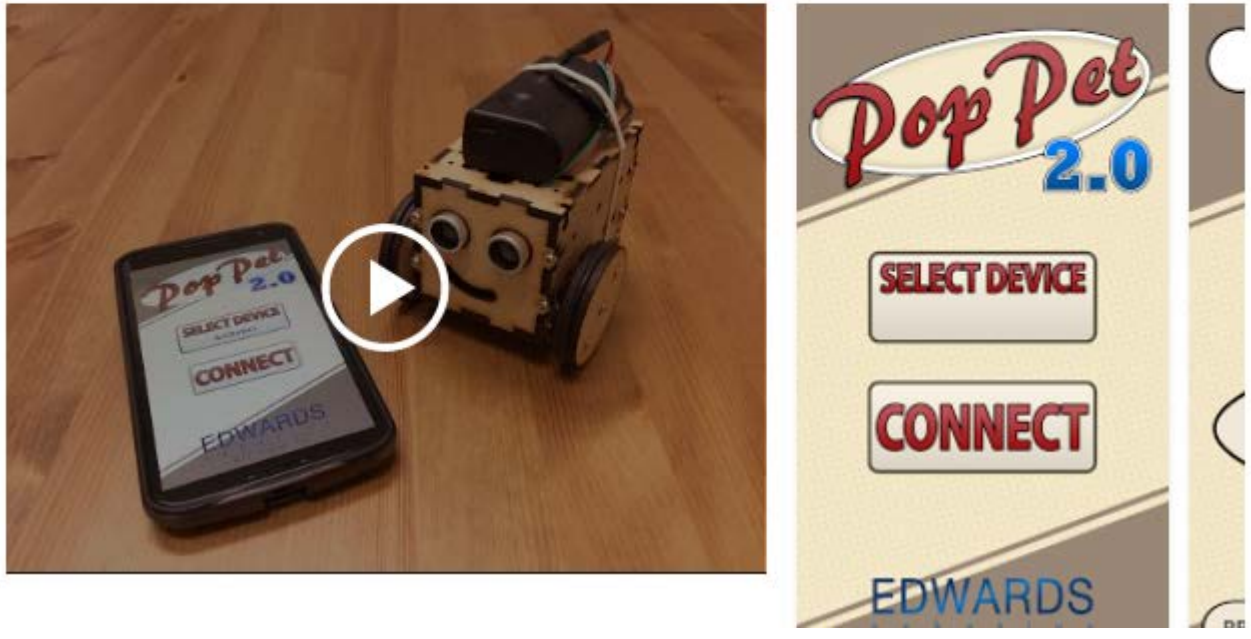
Sample 1: PopPet

After wiring your robot you just need to do these two steps.

Step 1: Install the free app on your Android based device

https://play.google.com/store/apps/details?id=com.cevinius.PopPet&hl=en





This app is completely free for you to download and use. It contains no ads or in-app purchases.

The app lets you:

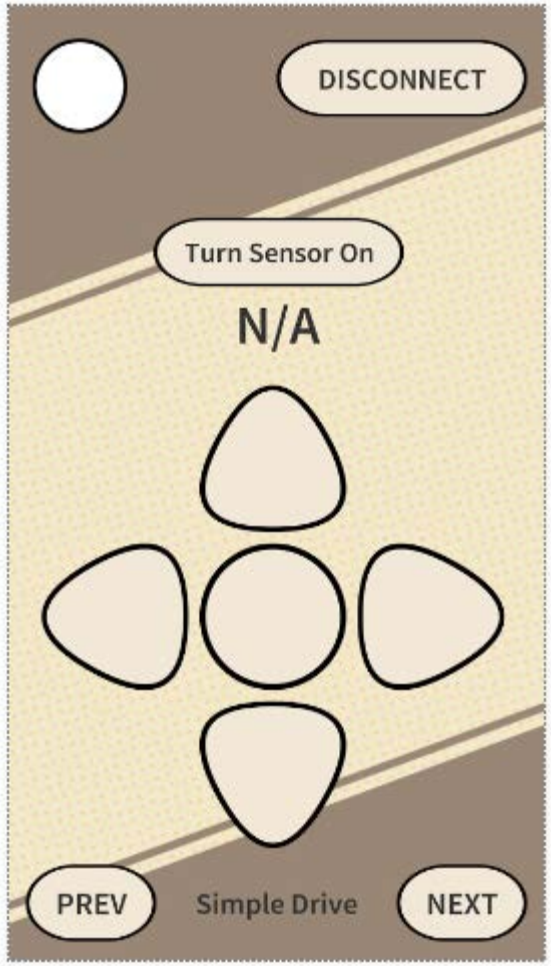
- Drive PopPet around using direction buttons
- Drive PopPet around using an analog stick
- Give PopPet voice commands (you can say go forward, go backwards, what time is it and others)
- Have PopPet drive around autonomously using your sonar sensor and explore

Step 2: You need to download the modified Arduino code for the V3 robot from http://www.roboticscity.com/ under Experiment 7

Note: before uploading this code to the robot you must disconnect the positive (+) connection to the Bluetooth module otherwise you will get an upload error. Once it uploads you can connect it.

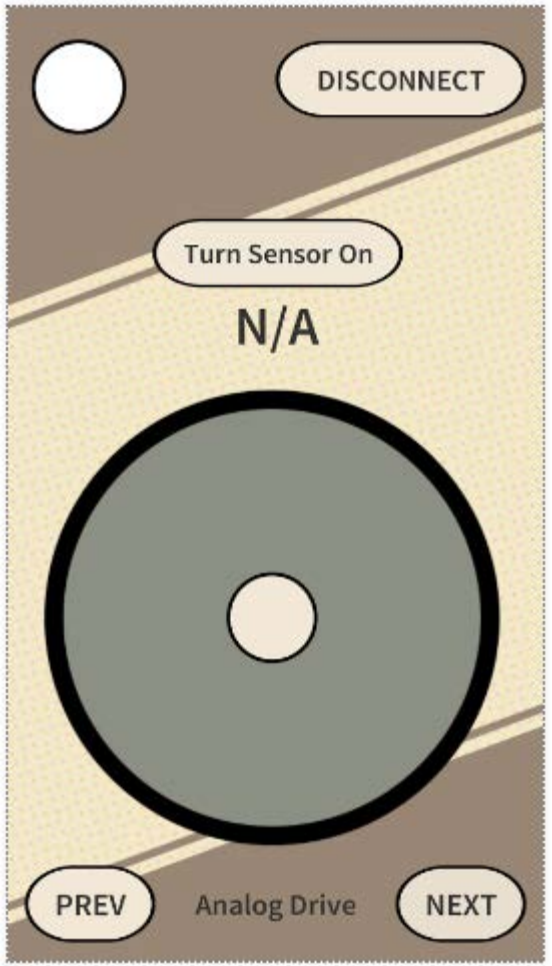
Sample 1: PopPet sample app screenshots

Once the robot components are connected, the App downloaded on your Android device and Arduino code uploaded to the robot make sure you pair up your robot to your phone.
Select the Bluetooth settings in your phone, turn on the V3 robot and scan fro BT devices from your phone. You will see the HC-06 show up and pair it. It may ask you for other prompts.
you can open the PopPet App and click on Select Device and pick the HC-06 device then click CONNECT.



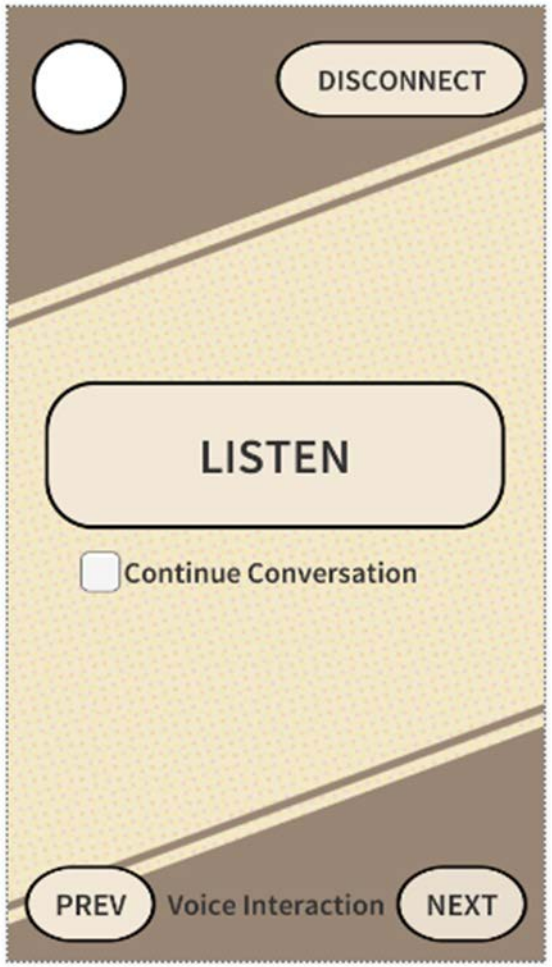
You can start navigating with the arrow, but you may need to modify the Arduino code to make this work exact. Instead click on NEXT to use the Analog pad to control the robot. See next page.

Sample 1: PopPet sample app screenshots



The Analog Drive gives you a better feel to control the robot. You can Turn Sensor On and get distance readings from the sonar sensor. Try the NEXT screen for voice control.
I am working on figuring out values that work better with the servos used in the V3 kit. The values this Arduino code provides are meant to be used with different motor controllers not the servos motor controllers.

Sample 1: PopPet sample app screenshots



Click LISTEN and say “move forward” or “move backwards” or “stop” or “what time is it” and it will respond and answer. Your phone must be connected to the Internet via cell or WiFi because the voice control actually goes out to a Google server, processes your commands and brings it back to the phone that then sends it to the Bluetooth module in your robot. For this reason there are times you will see a delay to reponses.

The app code is built upon the work by Jaidyn Edwards for his MobBob and Sphero Companion apps.Original Arduino code can downloaded from https://github.com/cevinius/PopPet/

PopPet is a robot kit developed by Jaidyn Edwards. http://www.poppettherobot.com/) This app was designed for use with PopPet 2.0, a Kickstarter robot, but it can be adapted to the Robotics++ V3 robot.

Explore mode uses the sonar sensor and can speak while exploring

A few notes about the Arduino code from : http://www.cevinius.com/2015/10/30/app-powered-poppet/

<DW,#,#> – Drive Wheels where the parameters are motor speeds for each wheel
<LW,#> – Set the speed of the left wheel only
<RW,#> – Set the speed of the right wheel only
Added commands for some of PopPet's Ultrasonic Sensor:
<UA,#> – Turns the Ultrasonic Sensor on/off
<US> – Requests an ultrasonic sensor reading.
Added a command for the explore mode.
<XA,#> – Turns explore mode on/off
While explore mode is activated, PopPet will send the following response codes to let the app know what's going on:
<XO> – Encountered an obstacle
<XC> – Changing direction just for fun (got bored going straight)
Initially, I didn't put the explore mode in the Arduino code. I first tried to do it all on the app. The app would receive regular sensor readings, and then adjust PopPet's course.
After some testing, I found that doing it that way had too much lag… Sometimes PopPet would hit the wall before he received the commands to back up and turn. I'm pretty confident the lag can be reduced by tweaking some of my delays between sending/receiving commands. (I'm probably overly paranoid about not spamming the serial port with commands… I think it can handle much more than what I'm throwing at it.) If I play with these settings, an app-side explore mode should work fine.
However, I decided to change tactics and put the explore mode into the Arduino-side code. This has a few advantages:
•PopPet can continue to explore even if you turn off your phone (after you've started the explore mode)
•The app is freed from driving and can do other stuff! While in Explore Mode, PopPet sends status codes <XO> and <XC> to the app to let it know what's going on. Since the app isn't busy driving PopPet, it has time to handle these status codes and react. In the current version, I have PopPet occasionally saying something (E.g. "Something's in my way", etc) in response to these codes. I found this can be a bit annoying, so I also added an option to turn off the explore mode narration!

Sample 2: RC Car

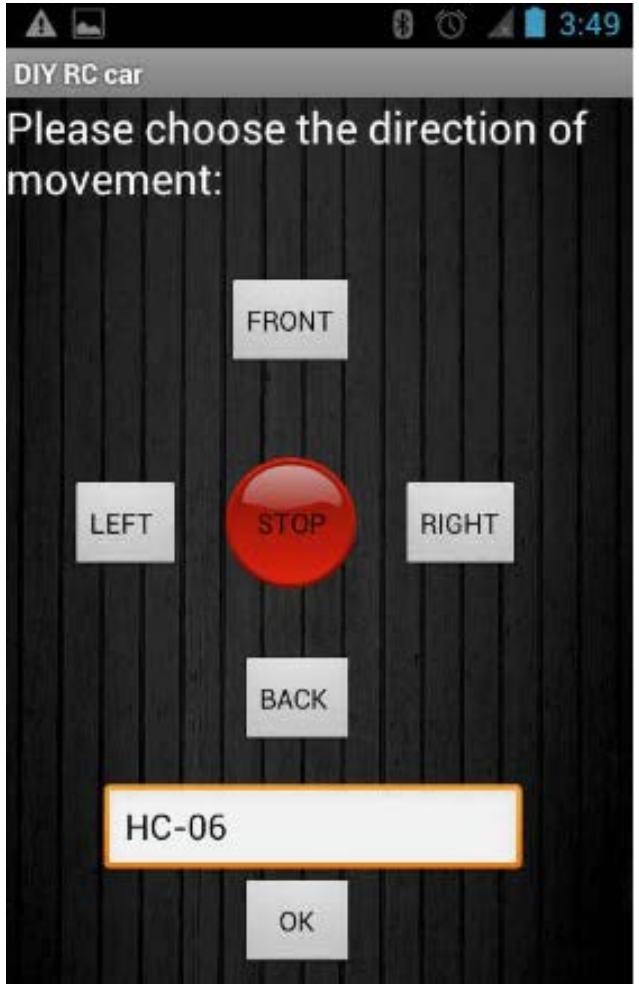
This next sample uses an App created in Eclipse with the Android Development Kit. The source code is available on the http://www.roboticscity.com/ under Experiment 7 Sample 2
Instructions for installing the Eclipse environment only if you wish to modify the source code for the app can be found here http://theopentutorials.com/tutorials/android/installing-android-sdk-and-eclipse-adt-plugin/ , but I do not recommended unless you are an advanced programmer.

In later examples I am going to show you how to make your own App very simply by using the MIT App Inventor and is very easy. Easier than Eclipse. What I like about this example is the simplicity of the App and Arduino source code.

The commands that are sent are in the form of ASCII. The arduino on the robot then checks the command received with it's previously defined commands and controls the servo motors depending on the command received to cause it to move forward, backward, left, right or to stop. Thus allowing us to create an android controlled robot.

Sample 2: RC Car

The commands that are sent are in the form of ASCII. The arduino on the robot then checks the command received with it's previously defined commands and controls the servo motors depending on the command received to cause it to move forward, backward, left, right or to stop. Thus allowing us to create an android controlled robot. Download app from DIYhacking.com and use modified Arduino code on the next slides.



```
#include <Servo.h>
Servo left;//Define left servo
Servo right;//Define right servo
byte val;

void setup()
{
  Serial.begin(115200);  // comm speed to BT module
  left.attach(9, 800, 2200); //left servo on digital pin 9 of arduino
  right.attach(10, 800, 2200); //right servo on digital pin 10 of arduino

}

void loop()
{
 int a=0;
 if(Serial.available())
  {
    val=Serial.read();
    Serial.println(int(val));

if(int(val)==49) //Move front
{
  right.write(180); //Rotates servo clockwise
  left.write(0); //Rotates servo anticlockwise
}
if(int(val)==50)   //Move back
{
  right.write(0);
  left.write(180);
}
```

```
if(int(val)==53)  //Stop
{
  right.write(90);//Stops the servos
  left.write(90);//Stops the servos
}

//or you can do it this way
//if(int(val)==53)  //Stop
 // {
 // left.writeMicroseconds(1500);
 // right.writeMicroseconds(1500);
 // }

 if(int(val)==51) //left
{
  right.write(180);
  left.write(90);
}
if(int(val)==52) //right
{
  right.write(90);
  left.write(0);
}

if(int(val)==55) //Extra button
{
  //Enter your code for any extra commands
} }    }
```