Arduino Powered Car with Mobile Control

A step by step Guide

Sakthi Dasan Sekar
shakthydoss@gmail.com

Preface

Purpose of this workshop is to extend the visibility of STG's Reach capability by building an embedded system from scratch and control it from mobile and wearable platforms. By completing this workshop you will be able to build a robotic car from scratch and write a remote control program for your smart phone to play around with the car.

Introduction to Embedded system

The first question that needs to be asked is "What exactly is an embedded computer?" An embedded computer is frequently a computer that is implemented for a particular purpose. In contrast, a modern PC computer usually serves a number of purposes: checking email, surfing the internet, listening to music, word processing, etc. In General embedded systems are systems that are programmed for a microcontroller (i.e. CPUs with integrated memory or peripheral interfaces) to perform a specific task forever.

Today, embedded systems are found in cell phones, digital cameras, camcorders, portable video games, calculators, and personal digital assistants, microwave ovens, answering machines, home security systems, washing machines, lighting systems, fax machines, copiers, printers, and scanners, cash registers, alarm systems, automated teller machines, transmission control, cruise control, fuel injection, anti-lock brakes, active suspension and many other devices/ gadgets.

Introduction to Arduino

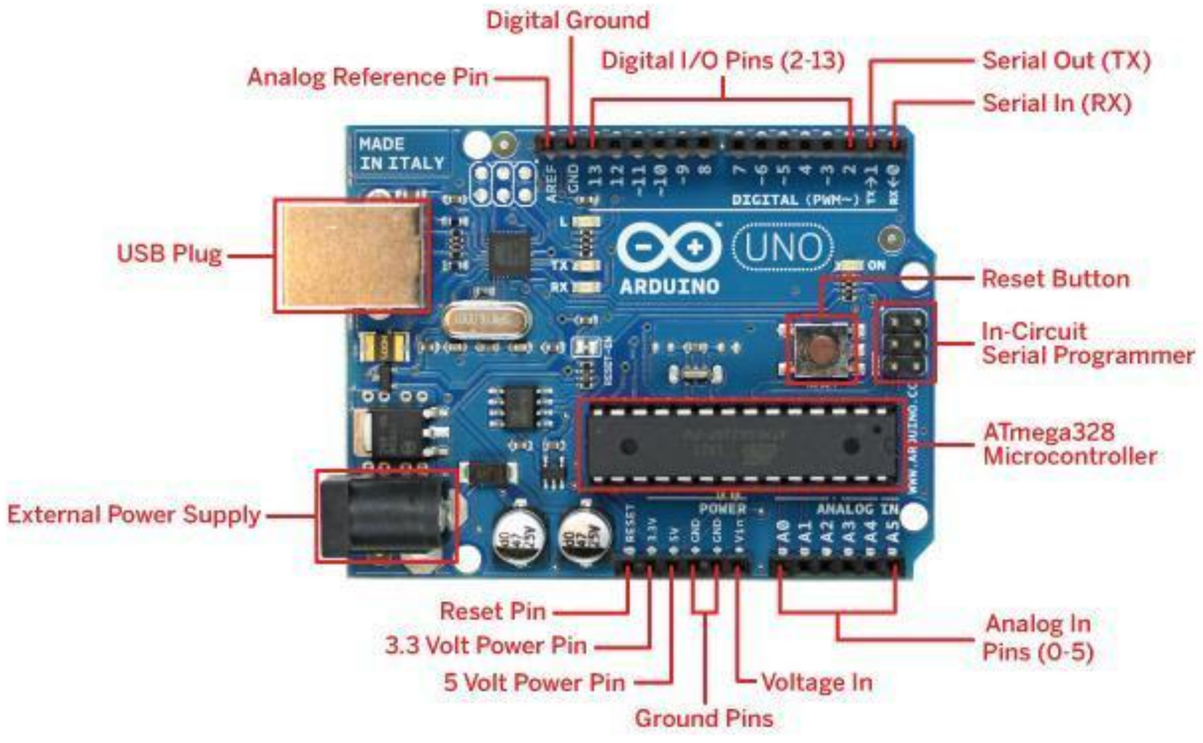
Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

ARDUINO
OPEN-SOURCE
COMMUNITY

Arduino can be used as interactive development kit such as taking inputs from a variety of switches or sensors, and controlling lights, motors, and other physical peripheral. Arduino projects can be stand-alone, or they can be controlled or communicate with software running on your computer, mobile or simple from another embedded device.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. And for this tutorial we will use Arduino board partially Arduino Uno for developing Bluetooth controlled car.

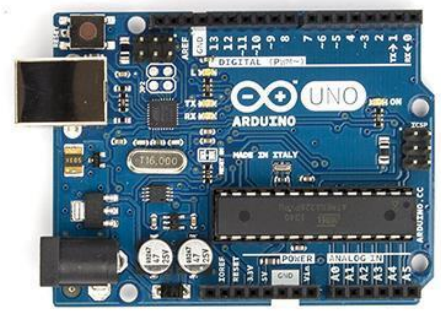
The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.



Hardware components

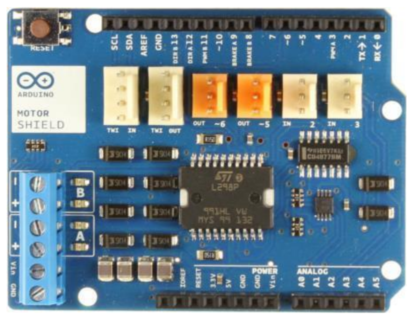
The Arduino board

The Arduino board will be the brain of the robot, as it will be running the software that will control all the other parts.



The motor driver

The Arduino Uno board cannot directly control a motor. The tricky part is to be able to make the motor selectively run forward or backwards, which requires swapping power and ground inputs into the motor. There is a specialized circuit called an H-Bridge (Arduino Motor Shield R3) that can do this, and there are several implementations of this circuit readily available for the Arduino platform.



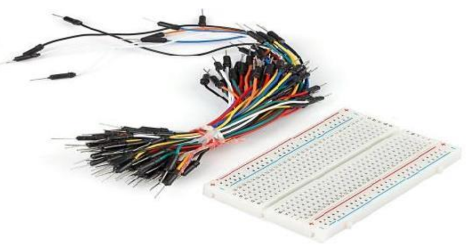
The Bluetooth Slave

The easiest way to control the car from a smartphone is to use the Bluetooth serial interface that is integrated modern smartphones. The phone will act as a

Bluetooth master, so we needed a Bluetooth slave (BT2S Bluetooth to Serial Slave) for the car to communicate with smartphones.

USB 2.0 A-Male to B-Male Cable, Prototyping Board and Cables

The Arduino board is connected to a computer via a USB port. The USB connection port is used to upload software and also can be used as a power source for Arduino board during testing. Since we indent to assemble and disassemble the car without soldering of hardware we also need breadboard board and connecting cables.



The Vehicle Kit (Vehicle Chassis)

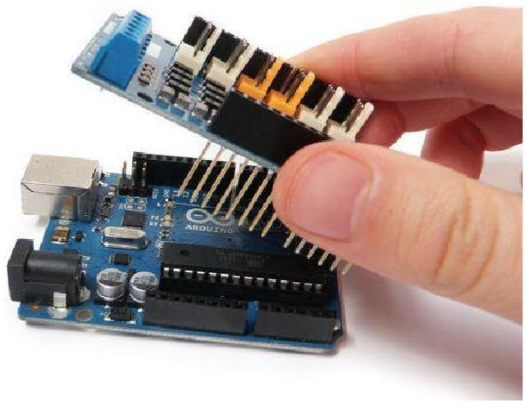
Vehicle Chassis provides physical structure and real estate to mount Arduino board and other hardware components. The Magician Chassis is a popular vehicle Chassis that is extremely simple to build. It includes two motors, two wheels, a balancer and a battery box that plugs directly into the Arduino board.

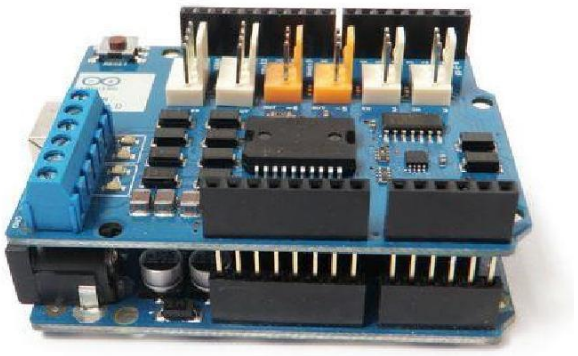
Circuit and Hardware Assembling.

Assembling Motor Shield

The Arduino Motor Shield allows you to easily control motor direction and speed using an Arduino. By allowing you to simply address Arduino pins, it makes it very simple to incorporate a motor into your project. It also allows you to be able to power a motor with a separate power supply of up to 12v.

The pins of the official Arduino motor shield will only align with Arduino Uno Rev. In order to make it work with older versions of the Arduino, you will need to trim a few pins off the motor shield. However, this is not, at all, recommended.





The motor shield has 2 channels, which allows for the control of two DC motors, or 1 stepper motor. It also has 6 headers for the attachment of

Arduino board inputs, outputs, and communication lines. The use of these pins is somewhat limited, and therefore not covered in this tutorial.

With an external power supply, the motor shield can safely supply up to 12V and 2A per motor channel (or 4A to a single channel).

There are pins on the Arduino that are always in use by the shield. By addressing these pins you can select a motor channel to initiate, specify the motor direction (polarity), set motor speed (PWM), stop and start the motor, and monitor the current absorption of each channel.

Arduino Motor Shield R3

Function	**Channel A**	**Channel B**
Direction	Digital 12	Digital 13
Speed (PWM)	Digital 3	Digital 11
Brake	Digital 9	Digital 8
Current Sensing	Analog 0	Analog 1

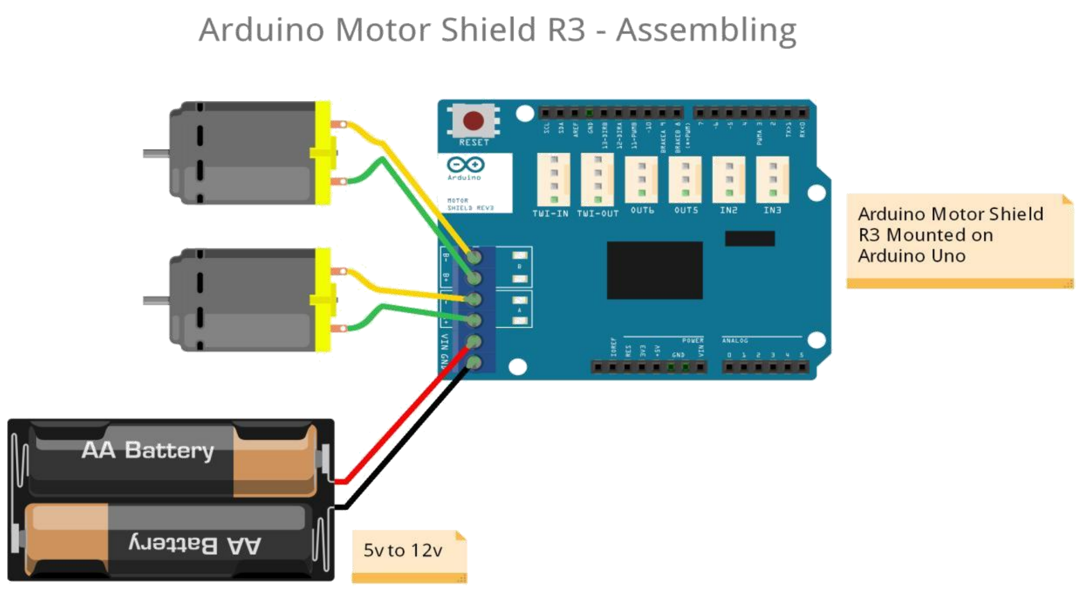
Plug the Arduino into your computer’s USB port and open the Arduino development environment. In order to get the board to do anything, you need to initialize the motor channel by toggling three parameters:

1. Set the motor direction (polarity of the power supply) by setting it either HIGH or LOW.

2. Disengage the brake pin for the motor channels by setting it to LOW.

3. Set the speed by sending a PWM command to the appropriate pin.

To control a motor using the Arduino Motor Shield, first plug the motor’s positive (red) wire into Channel A’s + terminal on the motor shield, and the motor’s ground (black) wire into Channel A’s – terminal on the shield.


Arduino Motor Shield R3 - Assembling
RESET
TWI-IN
TWI-OUT
OUT6
OUT5
IN2
IN3
Arduino Motor Shield R3 Mounted on Arduino Uno
AA Battery
AA Battery
5v to 12v

Similarly connect the other motor's positive (red) wire into Channel B's + terminal on the motor shield, and the motor's ground (black) wire into Channel B's – terminal on the shield.

An external power supply is not always necessary, but it drastically improves the motor's performance. It is recommended that you always use one.

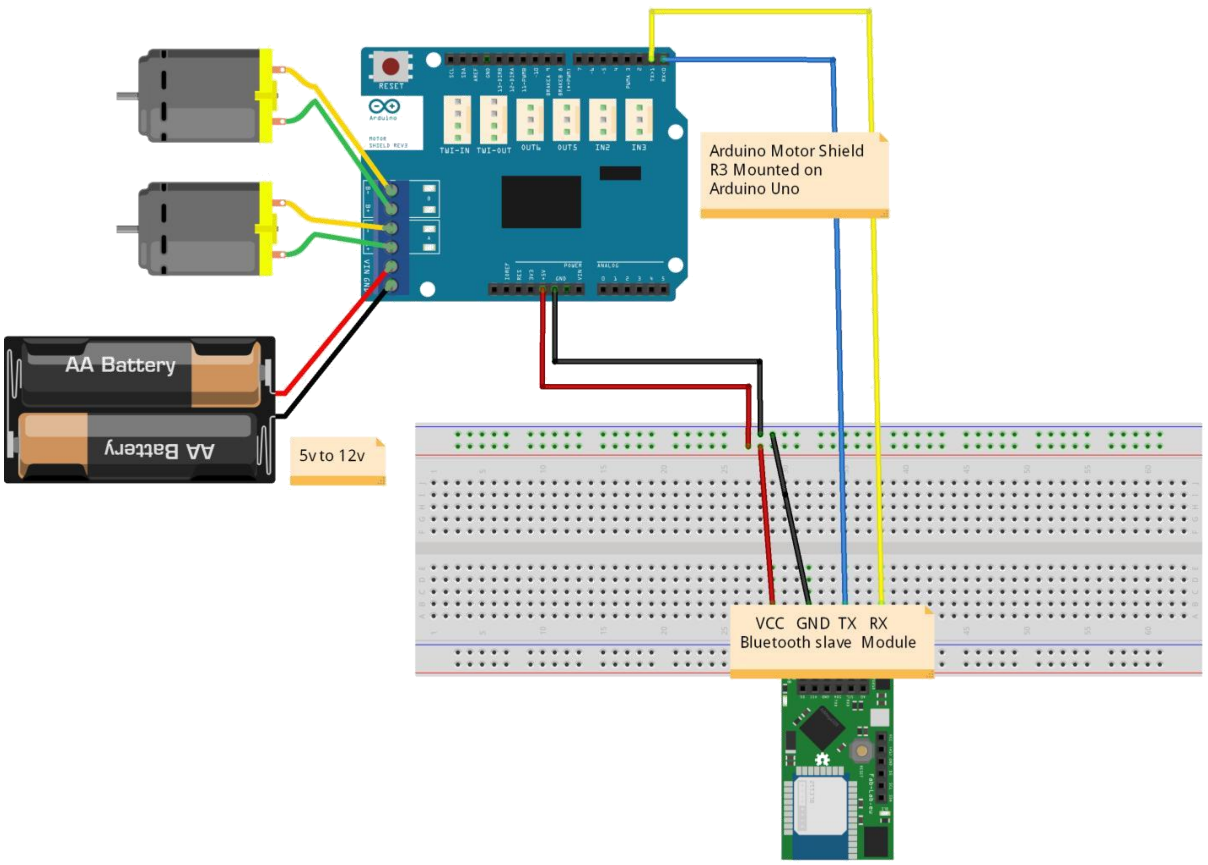
Assembling the Bluetooth

The vital piece of hardware that we intend to use in the robot vehicle is the Bluetooth slave, which connects to my smartphone to receive remote control commands.

The wiring of the Bluetooth slave to is pretty simple. The RXD and TXD pins in the Bluetooth slave go to digital pins 0 and 1 of Arduino Motor Shield which is mounted on Arduino Uno Board.

Bluetooth Shield (BT2S Bluetooth to Serial Slave) works over the standard serial communication protocols in the Arduino board. Thus when the Arduino board with Bluetooth is paired with a smartphone, a Serial.Print command via digital pin 1 will send the data to the phone. Similarly the Serial class also has functions to read data from the other end via the digital pin 0.

Arduino & Bluetooth slave - Assembling



Assembling the vehicle

The Magician Chassis vehicle kit includes plastic chassis, wheels, motors, battery box and screws. Assembling these parts should be very institutive and self-explanatory. All we have to do is to screw the plastic chassis with the wheels that are already attached with DC motors. Once done final assembled vehicle kit should look like the one shown in the picture.

Programming the Arduino

There are many varieties of Arduino boards that can be used for different purposes. Some boards look a bit different from the one below, but most Arduino have the majority of these components in common.

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply

The pins on your Arduino are the places where you connect wires to construct a circuit. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

GND (3): Short for 'Ground'. There are several GND pins on the Arduino, any of which can be used to ground your circuit.

5V (4) & 3.3V (5): As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.

Analog (6): A0 through A5 on the UNO are Analog In pins. These pins can read the signal from an analog sensor and convert it into a digital value that we can be read.

Digital (7): Across from the analog the digital pins 0 through 13 on the UNO can be used for both for sending and receiving inputs.

PWM (8): You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM).

AREF (9): Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

TX & RX

TX is short for transmit, RX is short for receive. Both are dedicated for serial communication.

Installing Arduino IDE

Go to the Arduino download page and download the latest version of the Arduino software (Arduino IDE) for your operating systems.

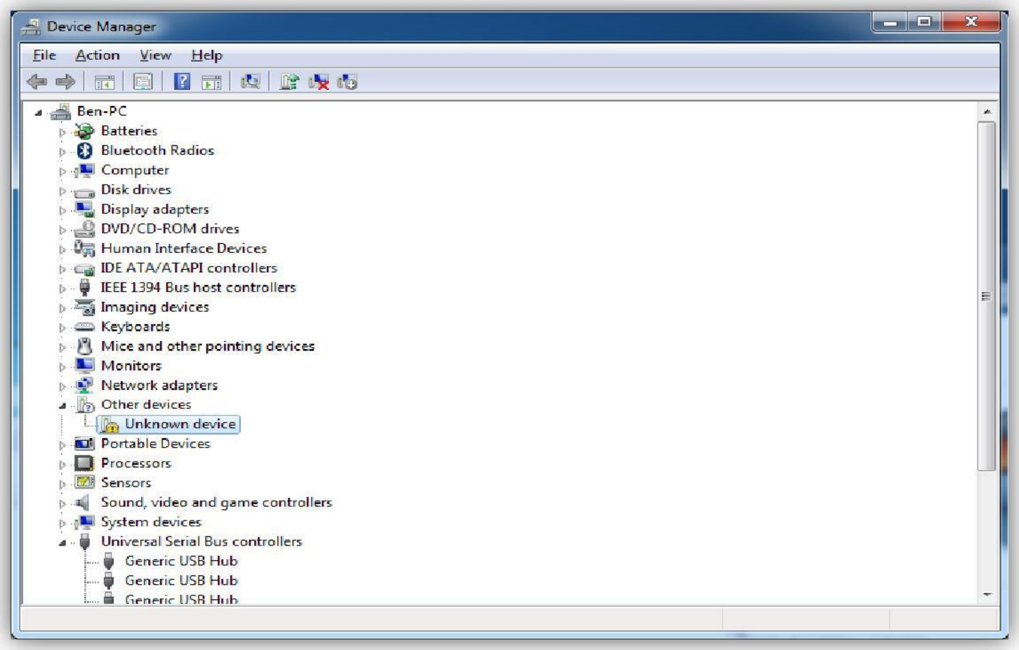
When the download is finished, un-zip it and open up the Arduino folder to confirm that yes, there are indeed some files and sub-folders inside. The file structure is important so don't be moving any files around unless you really know what you're doing. Power the Arduino by connecting the board to your computer with a USB cable. You should see the LED labeled 'ON' lights up.

Driver Setup (only for Windows)

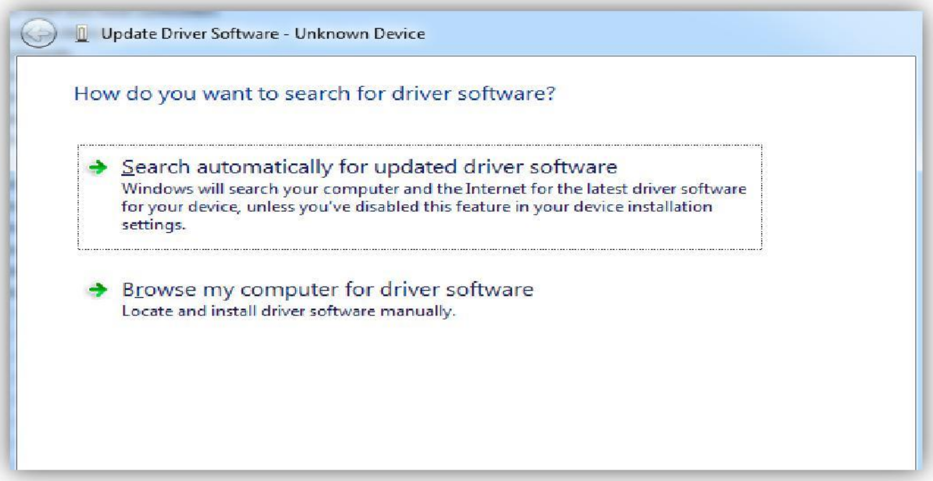
Installing the Drivers for the Arduino Uno (from Arduino.cc)

Plug in your board and wait for few moments.
Click on the Start Menu, and open up the Control Panel.
While in the Control Panel, navigate to System and Security. Next, click on System.

In System window is up, open the Device Manager. Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)". If there is no COM & LPT section, look under 'Other Devices' for 'Unknown Device'.



Right click on the “Arduino UNO (COMxx)” or “Unknown Device” port and choose the “Update Driver Software” option. Next, choose the “Browse my computer for Driver software” option.


Update Driver Software - Unknown Device

How do you want to search for driver software?

Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

Browse my computer for driver software
Locate and install driver software manually.

Finally, navigate to and select the Uno’s driver file, named “ArduinoUNO.inf”, located in the “Drivers” folder of the Arduino Software (you downloaded). If you cannot see the .inf file, it is probably just hidden. You can select the ‘drivers’ folder with the ‘search sub-folders’ option selected instead.

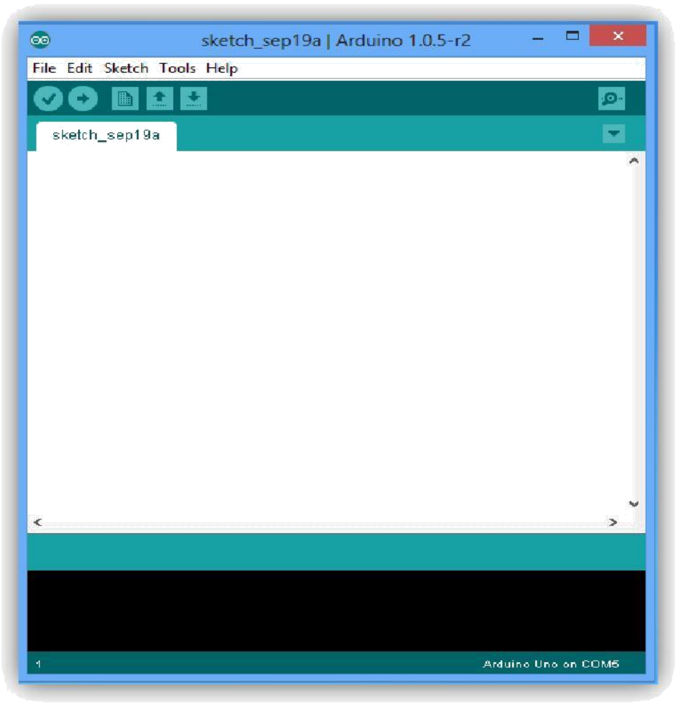
Programming the Arduino with Sketches

The white area is where the program is written. The programming language that the Arduino platform uses is called C++, a language that is regarded as very powerful but difficult to learn. C++ programs are written as regular text. Arduino calls these programs *Sketches*.

The black area at the bottom of the Arduino software window is where status information appears. For example, if your C++ code has an error this is where you will see the details.

The Arduino software also has a toolbar, with the following buttons

Verify: Runs the C++ compiler on your program. If there are any errors you will see them in the black status area.



Upload: Compiles your C++ program like the Verify button does, and if compilation is successful it uploads the resulting program to the Arduino board, which must be connected to the computer through the USB port.

New: Clears the program currently loaded.

Open: Shows a dropdown with a long list of example programs to load or the option to load a sketch from disk.

Save: Saves the current program to a file on disk.

Serial Monitor: Opens the serial monitor console. This is used mostly for debugging, as I will show in a little bit.

Programming Arduino for Bluetooth communication.

```
void setup()
{
    Serial.begin(9600);
    pinMode(13, OUTPUT);
}

void loop()
{
    if (Serial.available() > 0)
        { char ch = Serial.read();
        Serial.print("Received:
        "); Serial.println(ch);
        if (ch == '1') {
           // Motor run forward
        }
        if (ch == '2') {
           // Motor run backward
        }
    }
}
```

Programming Arduino Motor shield.

```
void loop(){
if (ch == '1') {
     // Forward control
     //Motor A forward at full speed
     digitalWrite(12, HIGH);    // Establishes forward direction
     digitalWrite(9, LOW);      // Disble the brake for Motor A
     analogWrite(3, maxspeed);  // Spins the motor on Channel A at full speed
     //Motor B backward at half speed
     digitalWrite(13, HIGH);    // Establishes forward direction
     digitalWrite(8, LOW);      // Disable the Brake for Motor B
     analogWrite(11, maxspeed); // Spins the motor on Channel B at half speed
 } else if(ch == '2'){
      // Backward  control
      //Motor A forward at full speed
      digitalWrite(12, LOW);    // Establishes backward direction
      digitalWrite(9, LOW);     // Disble the brake for Motor A
      analogWrite(3, maxspeed)   // Spins the motor on Channel A at full speed
      //Motor B backward at half speed
      digitalWrite(13, LOW);     // Establishes  backward direction
      digitalWrite(8, LOW);      // Disable the Brake for Motor B
      analogWrite(11,  maxspeed);// Spins the motor on Channel B at half speed
   }
}
```

Compile and upload the code to Arduino chip.

Next connect your Arduino board to your computer using the USB cable. With the Arduino board connected, make sure the correct model of Arduino board is selected by clicking Tool Menu $\rightarrow$ Board. Then in the Port submenu make sure serial port that is assigned to the Arduino connection.

Press the Upload button to compile and upload the sketch to the board. If there is anything wrong then appropriate error message will be displayed in black status bar.

If the Upload is successful the status area will show a message something similar to this

```
Binary sketch size: 1,978 bytes (of a 32,256 byte maximum)
```

Programming Android

In Android project we are going to write code for:

1. Open a Bluetooth connection
2. Send data
3. Listen for incoming data
4. Close the connection

But before we can do any of that we need to make sure Arduino and Android devices are paired. You can do this from the Android device in the standard way by Opening your application drawer and going to Settings. From there open Wireless and network. Then Bluetooth settings. From here just scan for devices and select Bluetooth device that is attached with Arduino board. If it asks for a pin enter 1234 to confirm authentication.

You shouldn't need to do anything special from the Arduino side other than to have it turned on. Next we need to tell Android that we will be working with Bluetooth by adding this element to the `<manifest>` tag inside the AndroidManifest.xml file:

```
<uses-permission android:name= "android.permission.BLUETOOTH" />
```

Now to get started we need a Bluetooth Adapter reference from Android. We can get that by calling

Now that we have the Bluetooth adapter and know that its turned on we can get a reference to our Arduino's Bluetooth device with this code:

```
Set pairedDevices = mBluetoothAdapter.getBondedDevices();
if(pairedDevices.size() > 0){
       for(BluetoothDevice device : pairedDevices){
       //you will need to change this to match the name of your device
             if(device.getName().equals("AUBTM-20")){
                   mmDevice =
                   device; break;
             }
      }
 }
```

Armed with the Bluetooth device reference we can now connect to it using this code:

```
//Standard SerialPortService ID
UUID uuid = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb"); mmSocket
= mmDevice.createRfcommSocketToServiceRecord(uuid); mmSocket.connect();

mmOutputStream = mmSocket.getOutputStream();
mmInputStream = mmSocket.getInputStream();
```

This opens the connection and gets input and output streams for us to work with. You may have noticed that huge ugly UUID. Apparently there are standard UUID’s for various devices and Serial Port’s use that one.

If everything has gone as expected, at this point you should be able to connect to your Arduino from your Android device. The connection takes a few seconds to establish so be patient. Once the connection is established the red blinking light on the Bluetooth chip should stop and remain off.

Closing the connection is simply a matter of calling close() on the input and output streams as well as the socket.

Now we are going to have a little fun and actually send the Arduino some data from the Android device. Make sure there is button click event that has code sent data to Arduino serial port buffer.

```
String msg = “1”; // sending message 1 to Arduino
device. mmOutputStream.write(msg.getBytes());
```

With this we have one way communication established! Make sure the Arduino is plugged in to the computer via the USB cable and open the Serial Monitor from within the Arduino IDE. Open the application on your Android device and open the Bluetooth connection. Now whatever sent from Mobile will be sent to the Arduino over air and magically show up in the serial monitor window. And by capturing the input received from serial port on Arduio board

Downloads

Arduino IDE http://arduino.cc/en/main/software

Android ADT http://developer.android.com/sdk/index.html

Arduino side Code https://github.com/stgishare/arduino-bluetooth-car/

Android Bluetooth Code. https://github.com/stgishare/arduino-bluetooth- car/

Reference

http://playground.arduino.cc/learning/ Learning Arduino & Getting Started
http://arduino.cc/en/Main/ArduinoBoardUno Schematic & Reference Design
http://arduino.cc/en/Main/ArduinoMotorShieldR3 Schematic & Reference Design

http://arduino.cc/en/Reference/HomePage Arduino programming API Reference http://42bots.com/tutorials/how-to-connect-arduino-uno-to-android-phone- via-bluetooth/ JY-MCU module communicates via serial connection http://arduino.cc/en/Tutorial/DueMotorShieldDC Motor Shield and DC motor control https://learn.sparkfun.com/tutorials/using-the-bluesmirf/example-code- using-command-mode Using Command Mode
http://arduino.cc/en/Main/Software Arduino Software Downloads
http://developer.android.com/guide/topics/connectivity/bluetooth.html
Android Bluetooth API