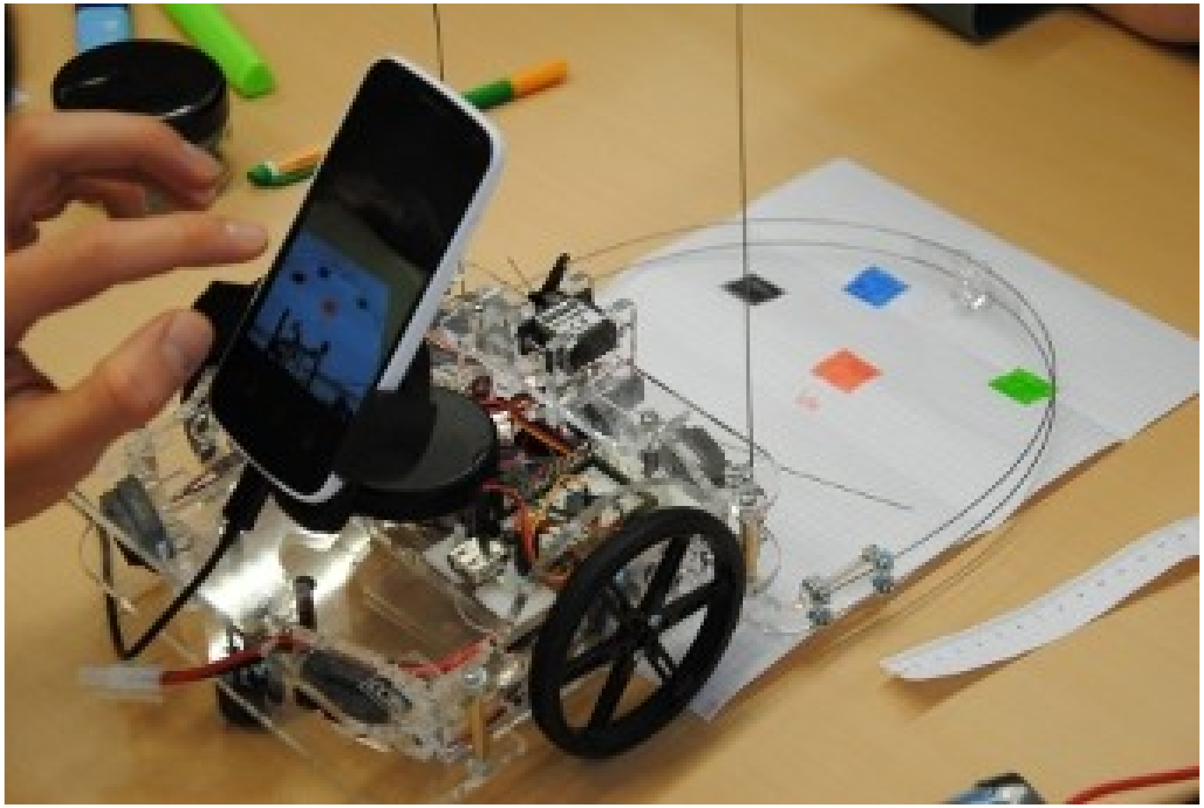
Robot Introduction



Questions:
alexander.hirsch@student.uibk.ac.at

Content

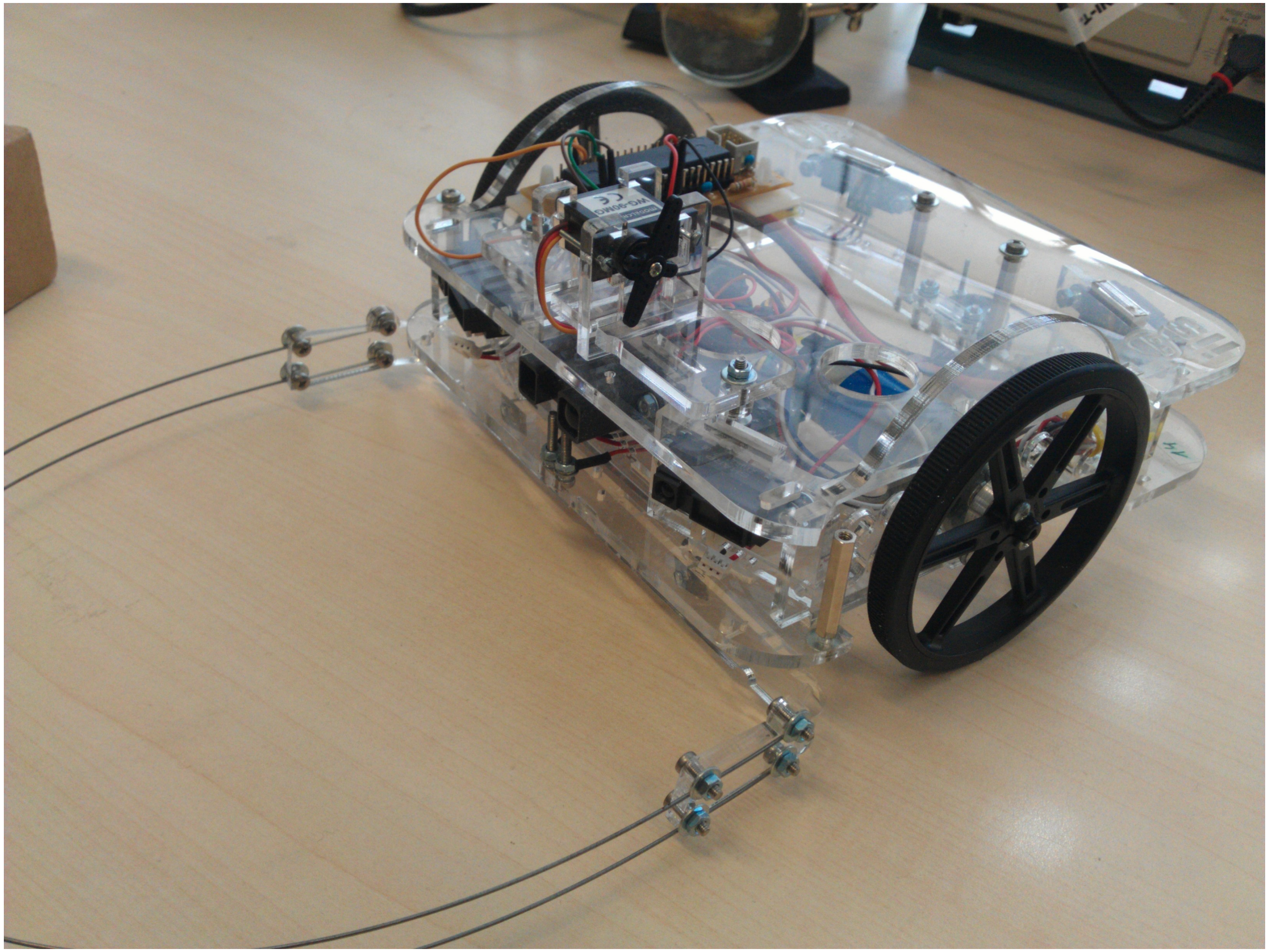
- The Robot (internals + components)
- Communication (USB OTG / Bluetooth)
- Setup everything
- Example App
- Details:
  I2C, H-Bridge, PWM, Sensors

My Experiance (last year)

Now Better (hopefully)

- Improved communications
- Code examples
- Knowledge about firmware / interior setup
- Not multithreaded (better debugging)
- Easy to use interface

The Robot



2 Layers

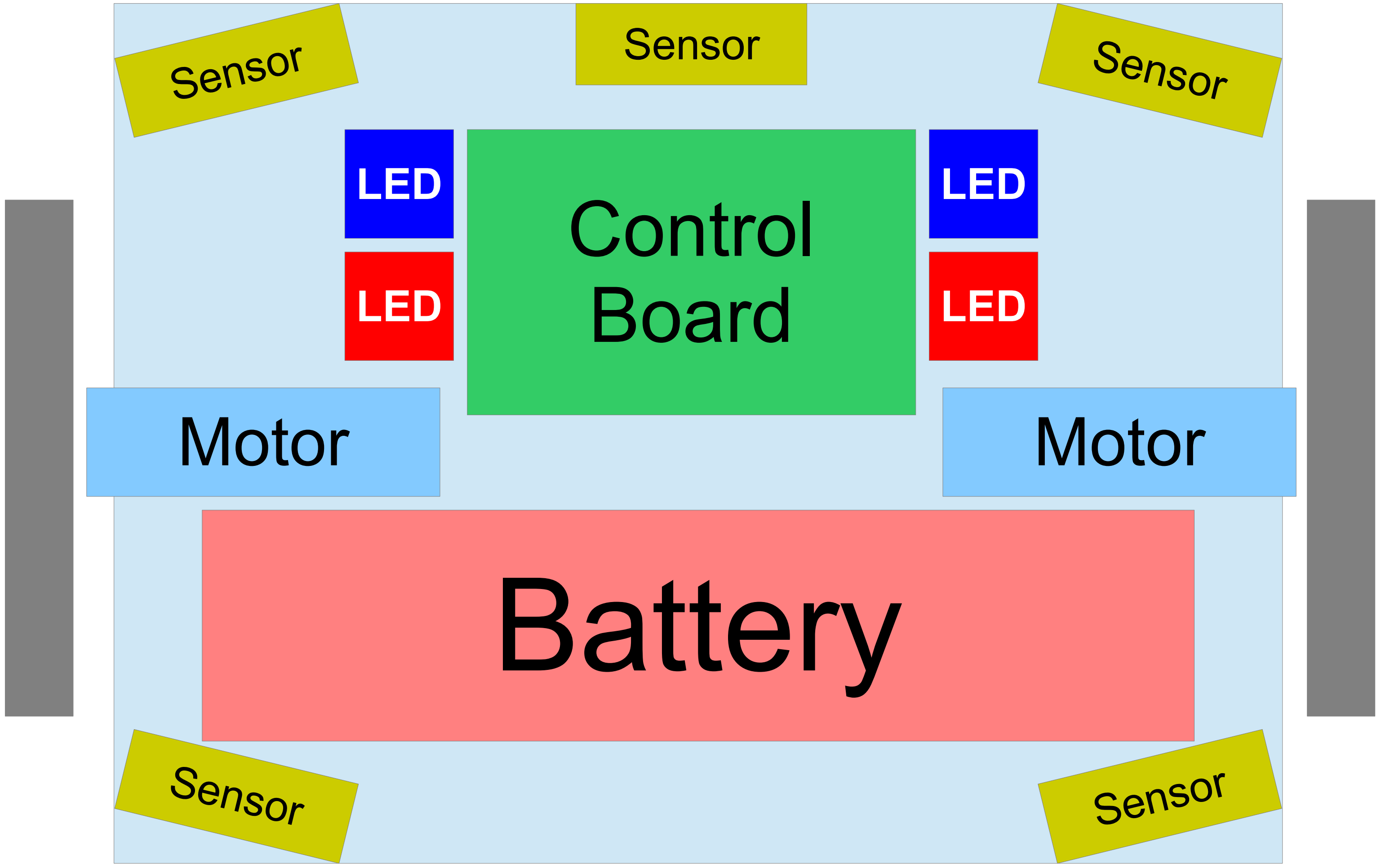
Interior

- DC Motors (movement)
- IR Sensors (range detection)
- Control Board
- Battery

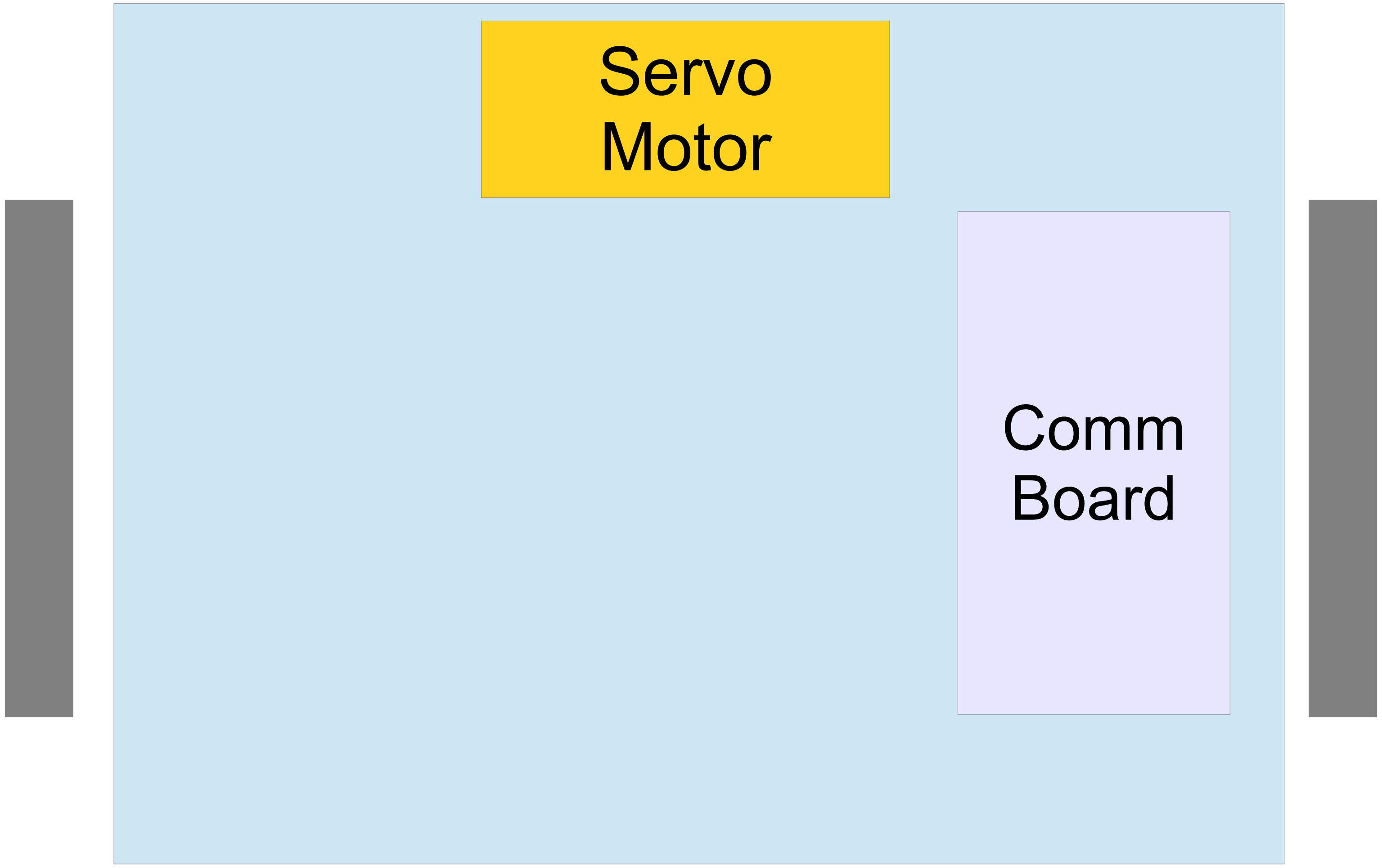
On Top

- Comm Board
  → ***new*** ←
- Servo Motor (Catching Balls)

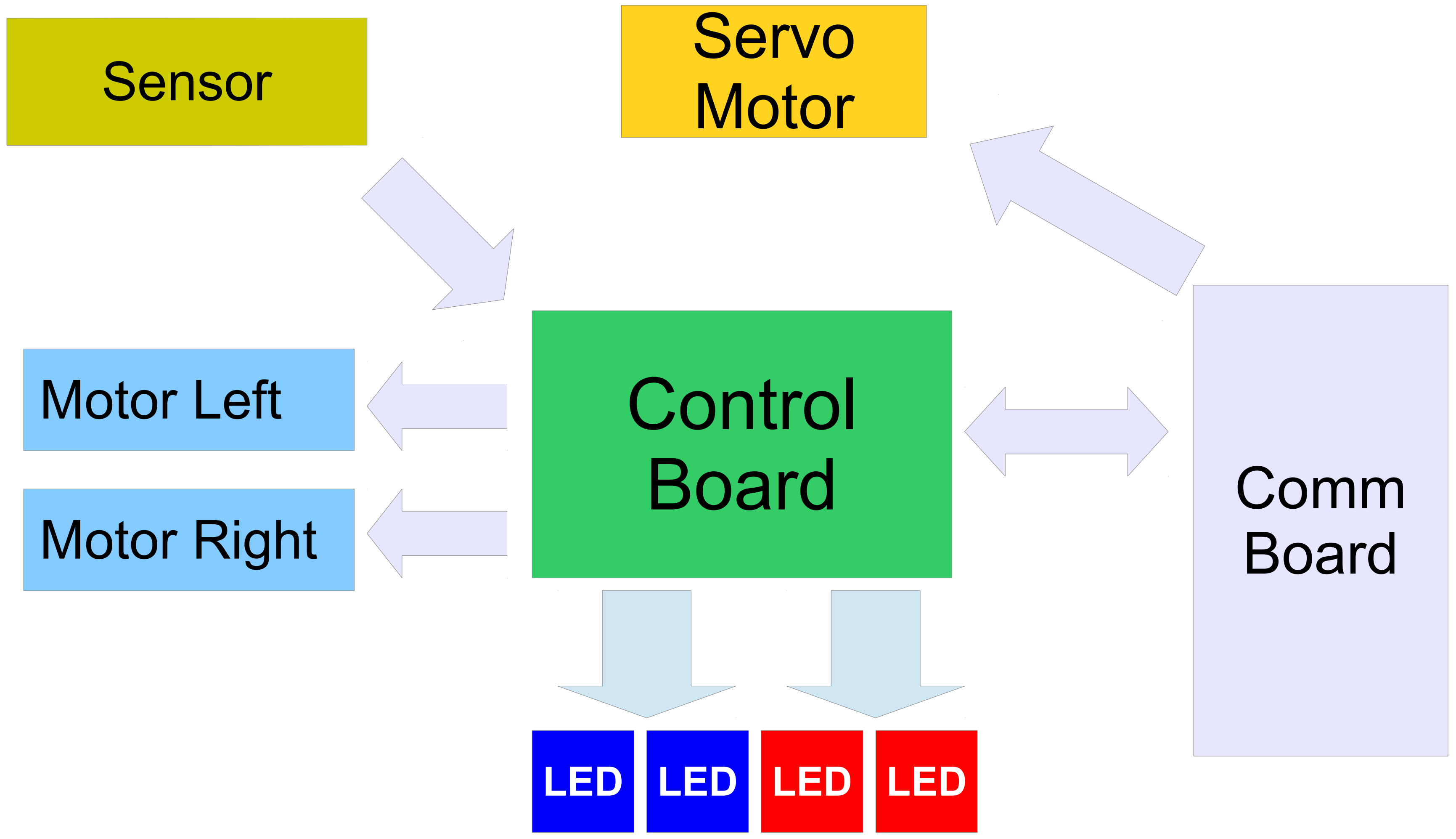
Interior


Sensor
Sensor
Sensor
LED
LED
Control Board
LED
LED
Motor
Motor
Battery
Sensor
Sensor

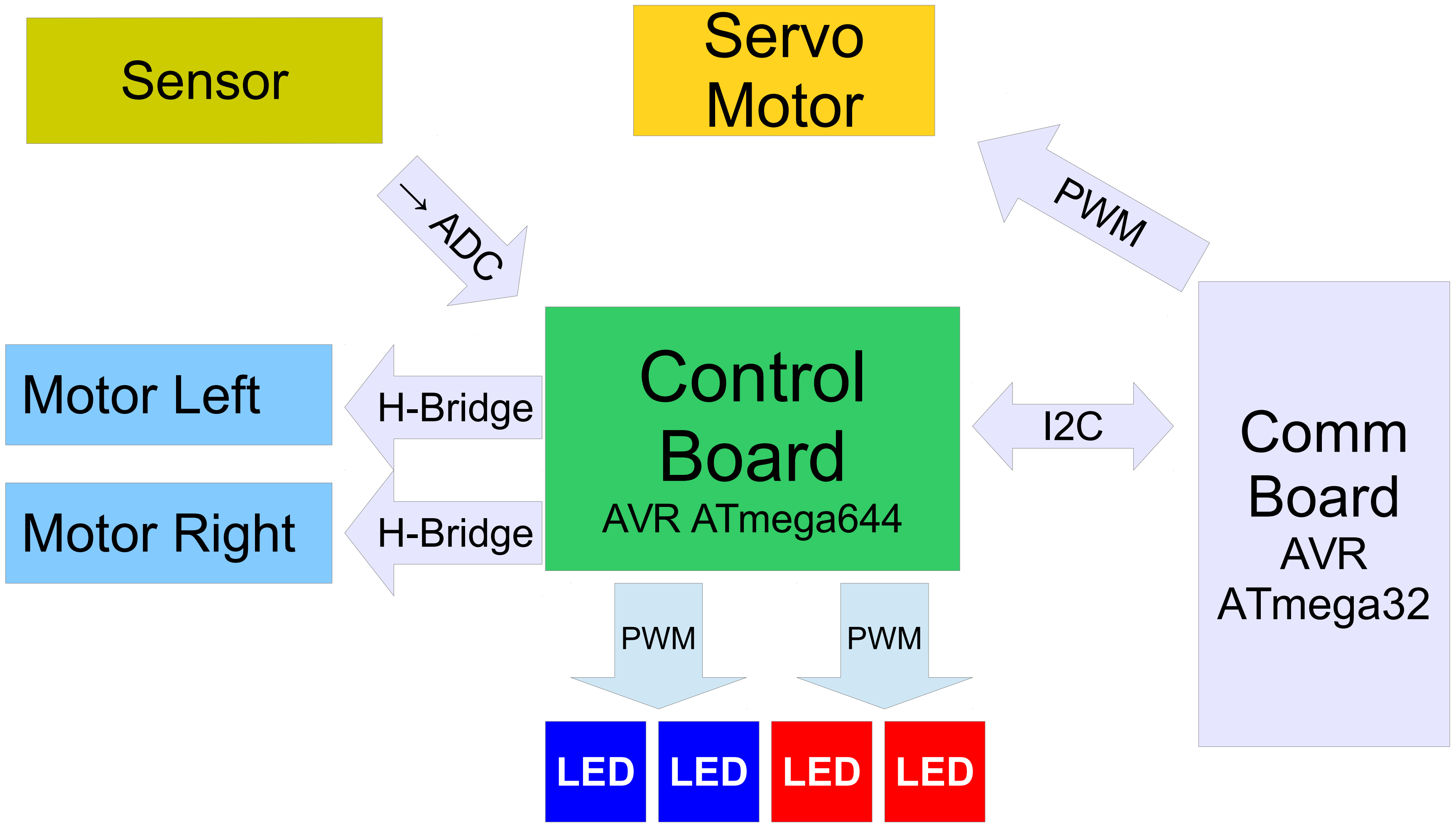
On Top



Component Interaction

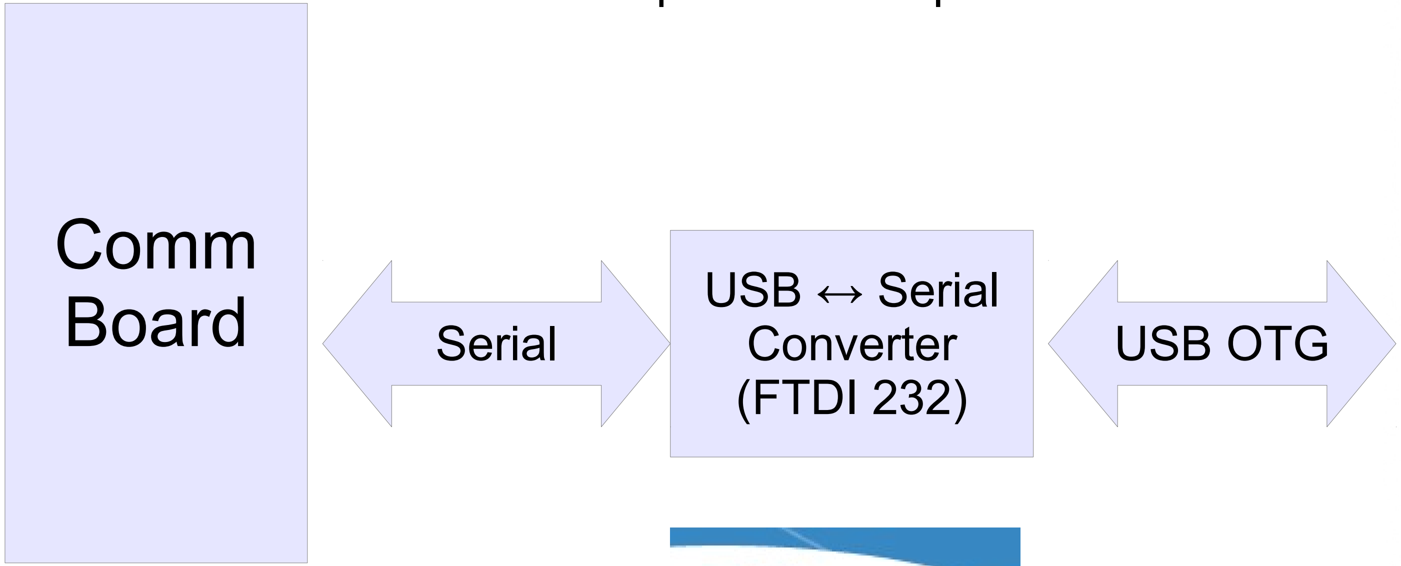


Component Interaction (detail)



Communication (wired)

Benefits of Serial Connection:
- very easy compared to USB
- device / platform independent

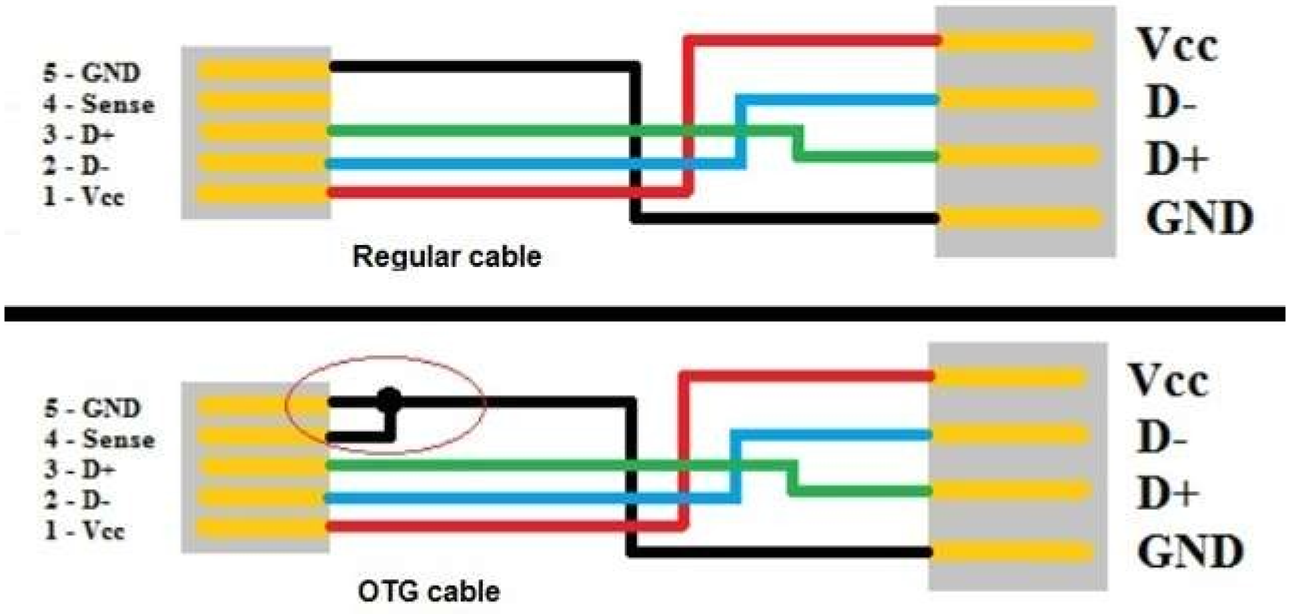


FTDI Chip

Code Examples
Provided, scroll down

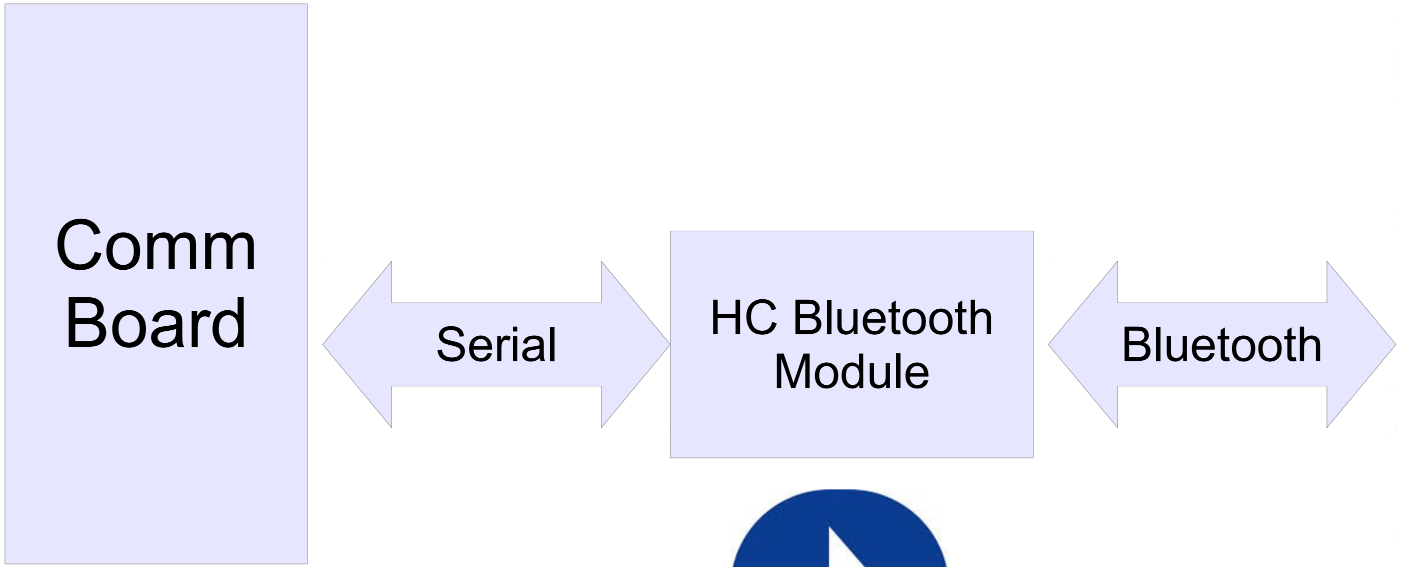
Phone powers Converter + Comm Board

USB OTG (on the go)



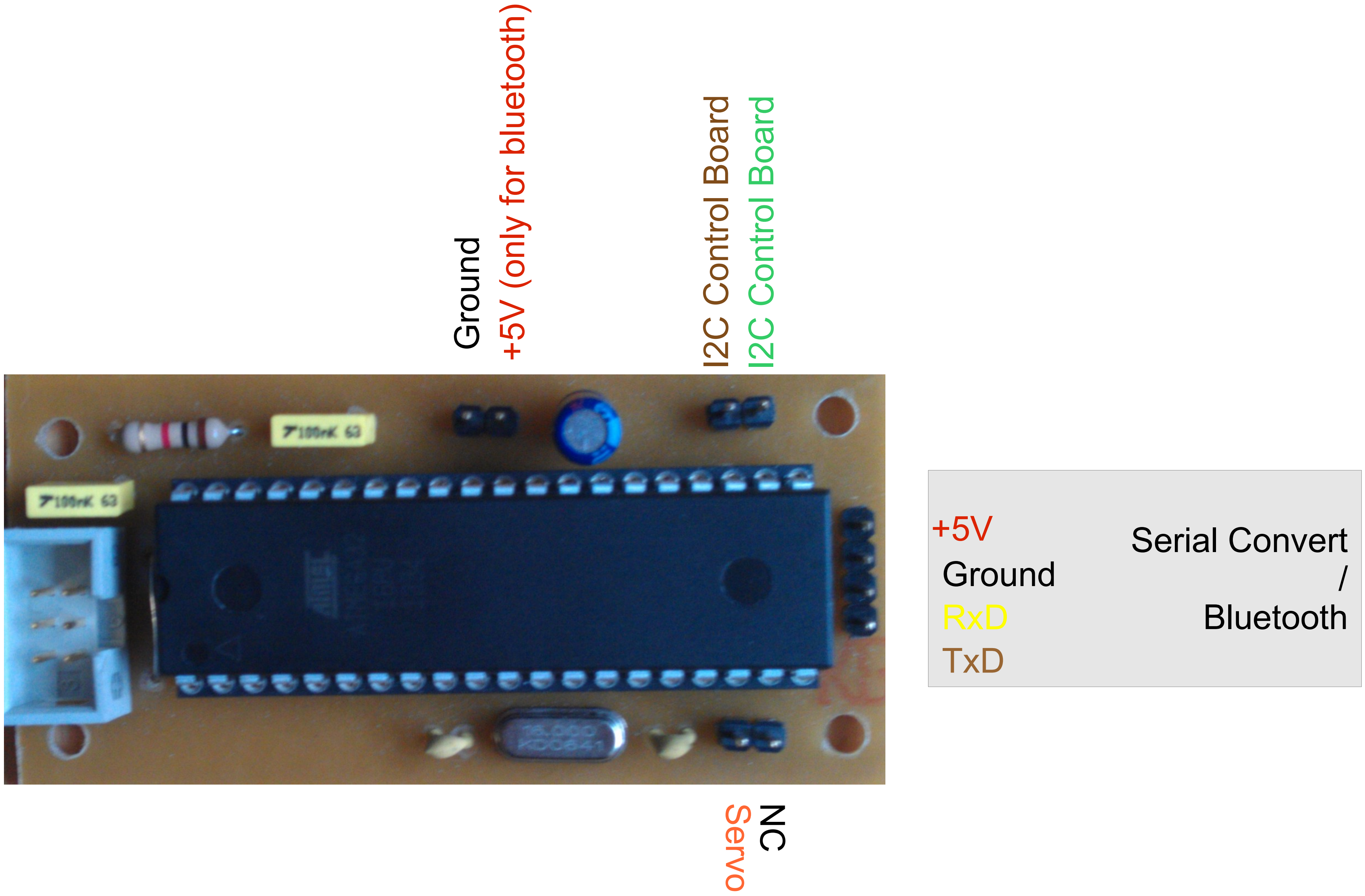
Communication (wireless)

Comm Board Powers Bluetooth module



Code Examples
→ Google

Pair w/ Phone
Secret code: 1234

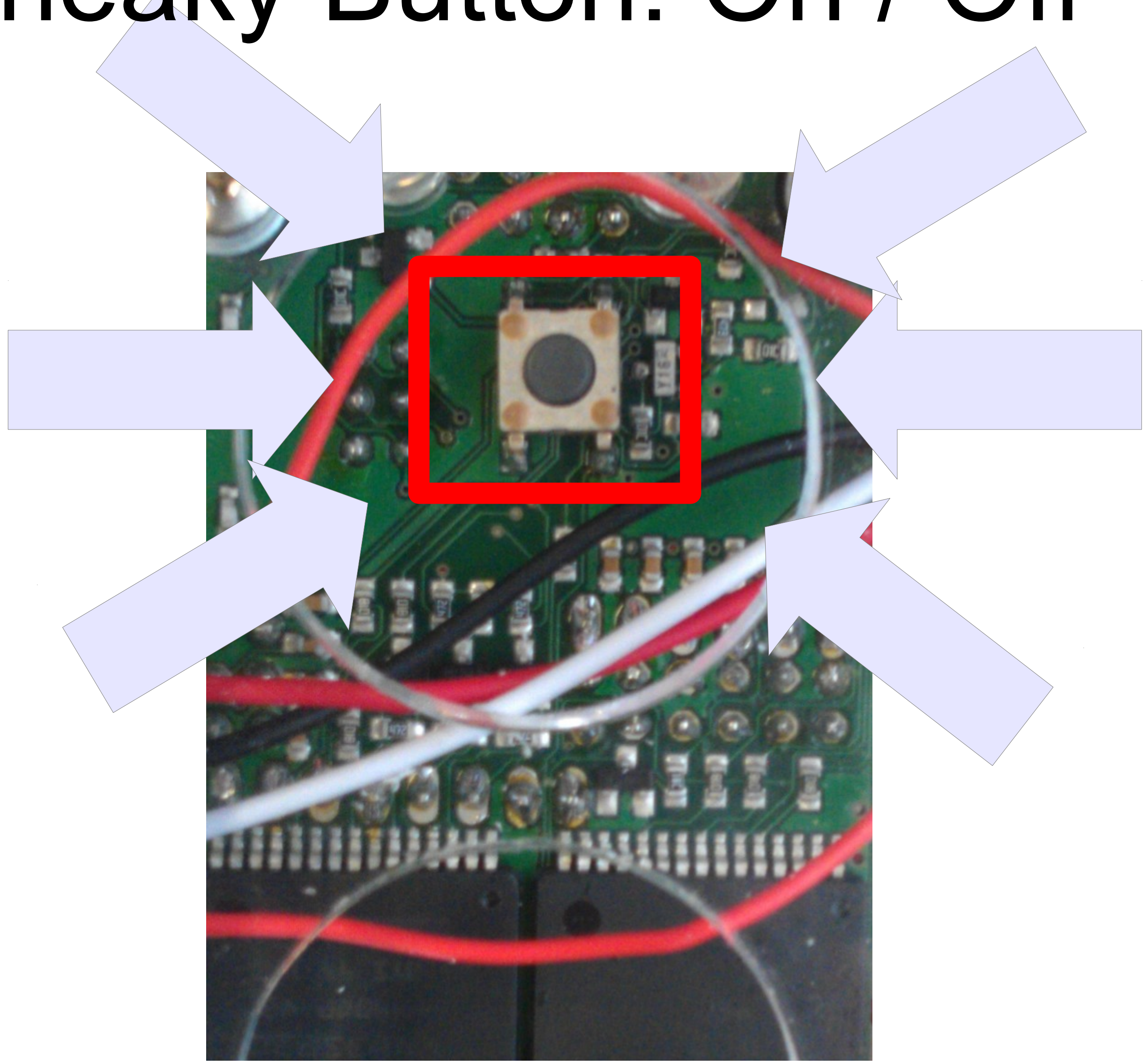


NC … not connected

Setup everything up

1) Make sure Comm Board is connected propperly
2) Connect serial converter / Bluetooth module to Comm Board
3) Check if battery is connected
4) Push button on the <u>underside</u> of the Robot
5) Connect Phone

Sneaky Button: On / Off

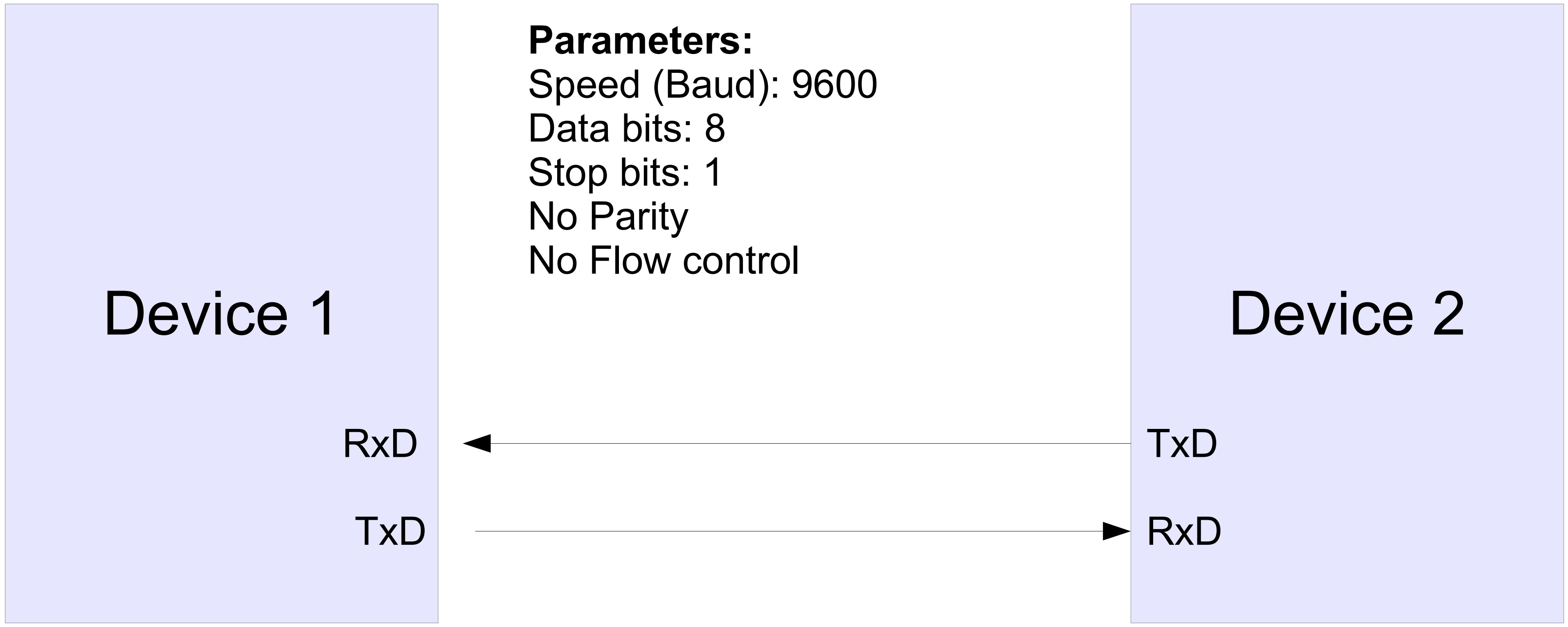


How does Serial Work

Detail:
Same as RS232
but with 0V 5V

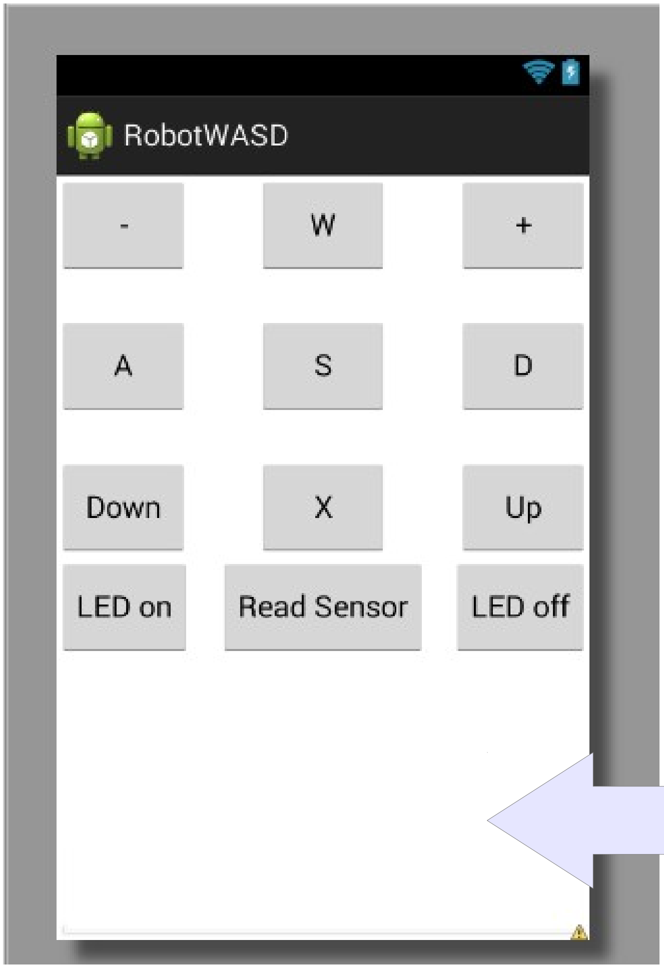
It's character based!

Parameters:
Speed (Baud): 9600
Data bits: 8
Stop bits: 1
No Parity
No Flow control



Alex Hirsch

WASD example App



W	Move forward
S	Stop
A	Turn left
D	Turn right
X	Move backward
-	Lower bar a few degree
+	Rise bar a few degree
Down	Fixed position for bar (low)
Up	Fixed position for bar (high)

Debug Output + Sensor Data Below

Application needs FTDriver library
https://github.com/ksksue/FTDriver

WASD Code FTDriver instance

```
private FTDriver com;

@Override
protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	setContentView(R.layout.activity_main);

	textLog = (TextView) findViewById(R.id.textLog);

	com = new FTDriver((UsbManager) getSystemService(USB_SERVICE));

	connect();
}
```

WASD Code Write Data

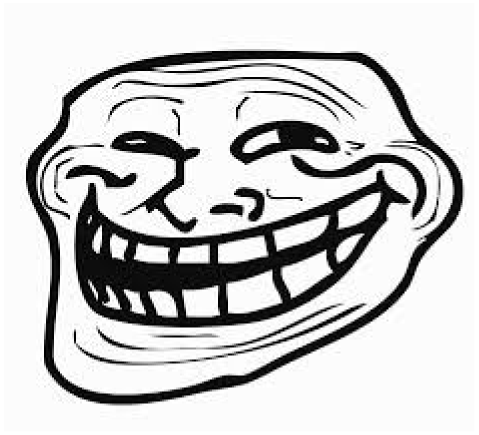
```
private FTDriver com;

public void comWrite(byte[] data) {
	if (com.isConnected()) {
		com.write(data);
	} else {
		textLog.append("not connected\n");
	}
}
```

WASD Code Read Data

```
private FTDriver com;

public String comRead() {
        String s = "";
        int i = 0;
        int n = 0;
        while (i < 3 || n > 0) {
                byte[] buffer = new byte[256];
                n = com.read(buffer);
                s += new String(buffer, 0, n);
                i++;
        }
        return s;
}
```



Strange android tripple buffering

WASD Code ReadWrite Data

```
private FTDriver com;

public String comReadWrite(byte[] data) {
        com.write(data);
        try {
                Thread.sleep(100);
        } catch (InterruptedException e) {
                // ignore
        }
        return comRead();
}
```

WASD Code Commands

Commands without parameter

w	Move forward
s	Stop
a	Turn left
d	Turn right
x	Move backward
-	Lower bar a few degree
+	Rise bar a few degree
r	LEDs on
e	LEDs off
q	Read sensors

Example:
new byte[] {'w', '\r', '\n'};

Commands with parameter

```
public void robotSetLeds(byte red, byte blue)
{
        ComReadWrite(
                new byte[] { 'u', red, blue, '\r', '\n' }
        );
}

public void robotSetVelocity(byte left, byte
right) {
        ComReadWrite(
                new byte[] { 'i', left, right, '\r', '\n' }
        );
}

public void robotSetBar(byte value) {
        ComReadWrite(
                new byte[] { 'o', value, '\r', '\n' }
        );
}
```

\r is ignored anyway, \n is used as lineending

Android USB Permission (simple)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ...>
      ...
   <uses-feature android:name="android.hardware.usb.host" />
   <application ...>
      <activity ...>
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
            <action
android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />
         </intent-filter>
         <meta-data
            android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
            android:resource="@xml/device_filter" />
      </activity>
   </application>
</manifest>
```

Android USB Permission (better)

- Use intent filter

  +

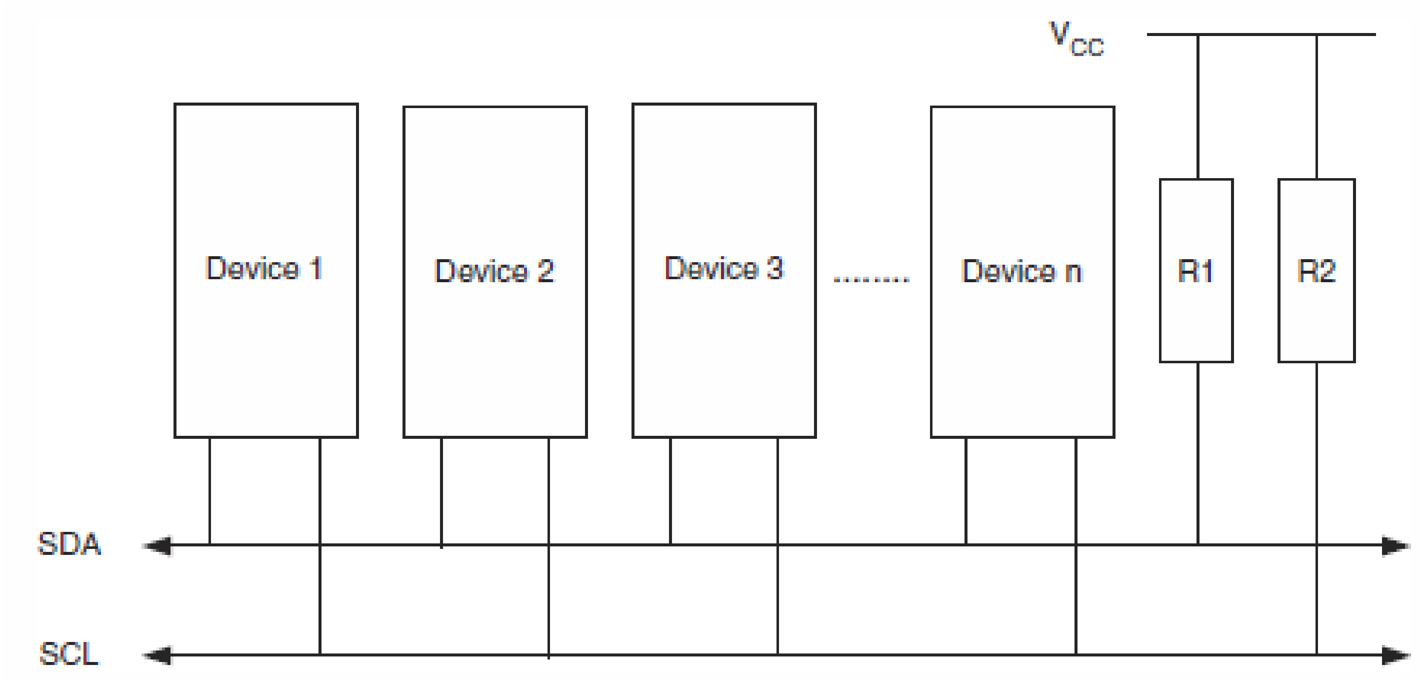
- Implement Permission request

  see https://developer.android.com/guide/topics/connectivity/usb/host.html

Bluetooth

- see http://luugiathuy.com/2011/02/android-java-bluetooth/
- see http://english.cxem.net/arduino/arduino5.php

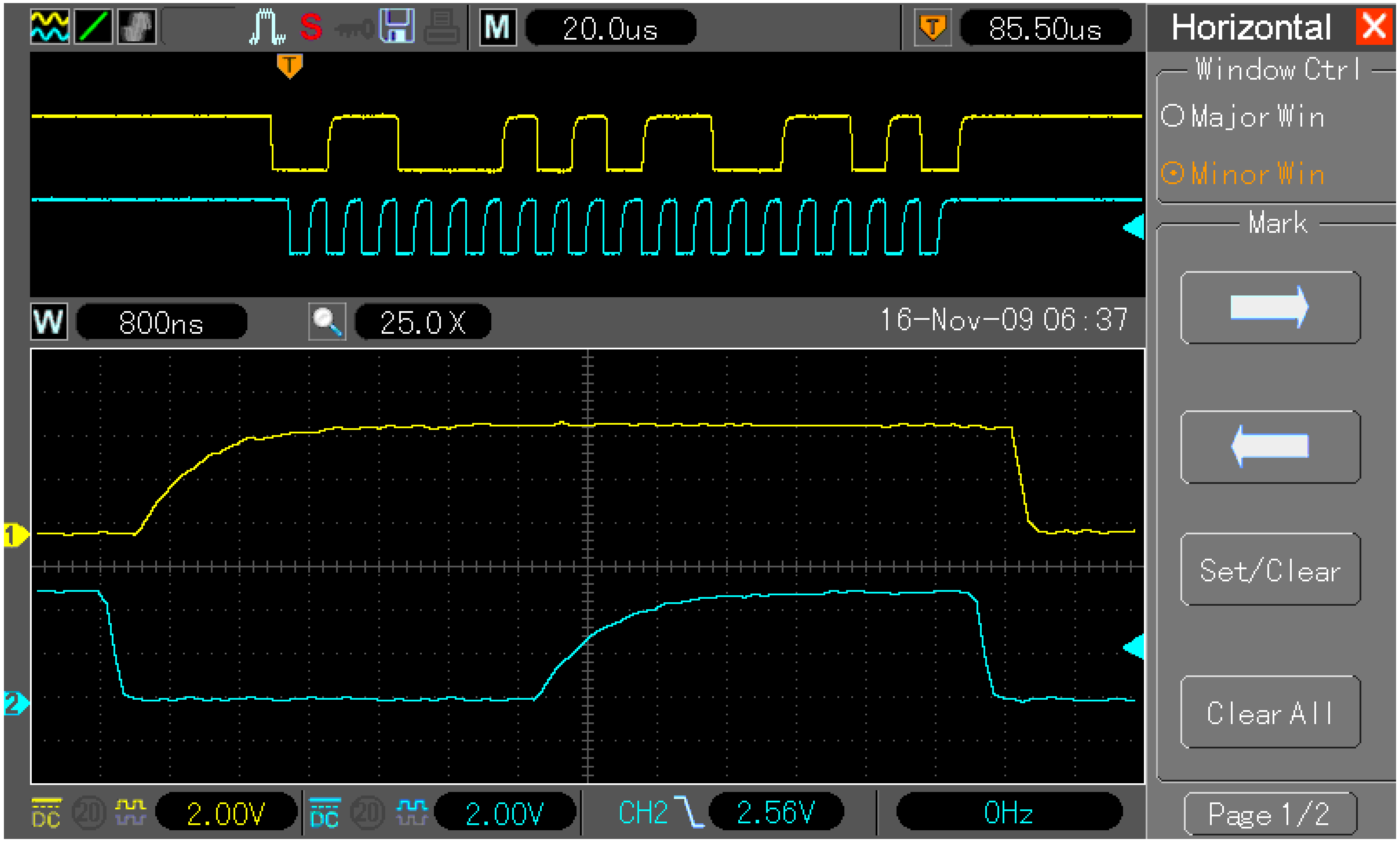
I2C Communication



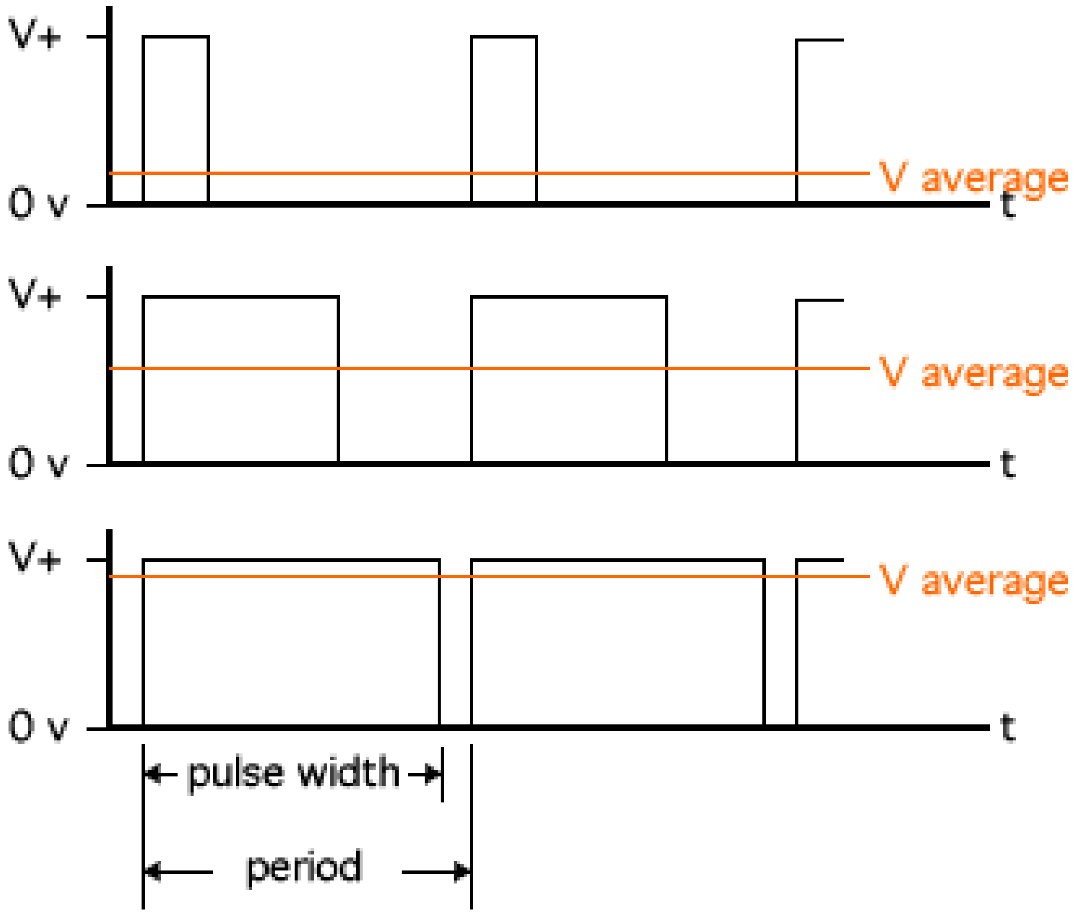
Master / Slave
Each slave has a 7 Bit address
+ 1 bit to state read or write access

Src: ATmega32 datasheet

I2C Communication (example)

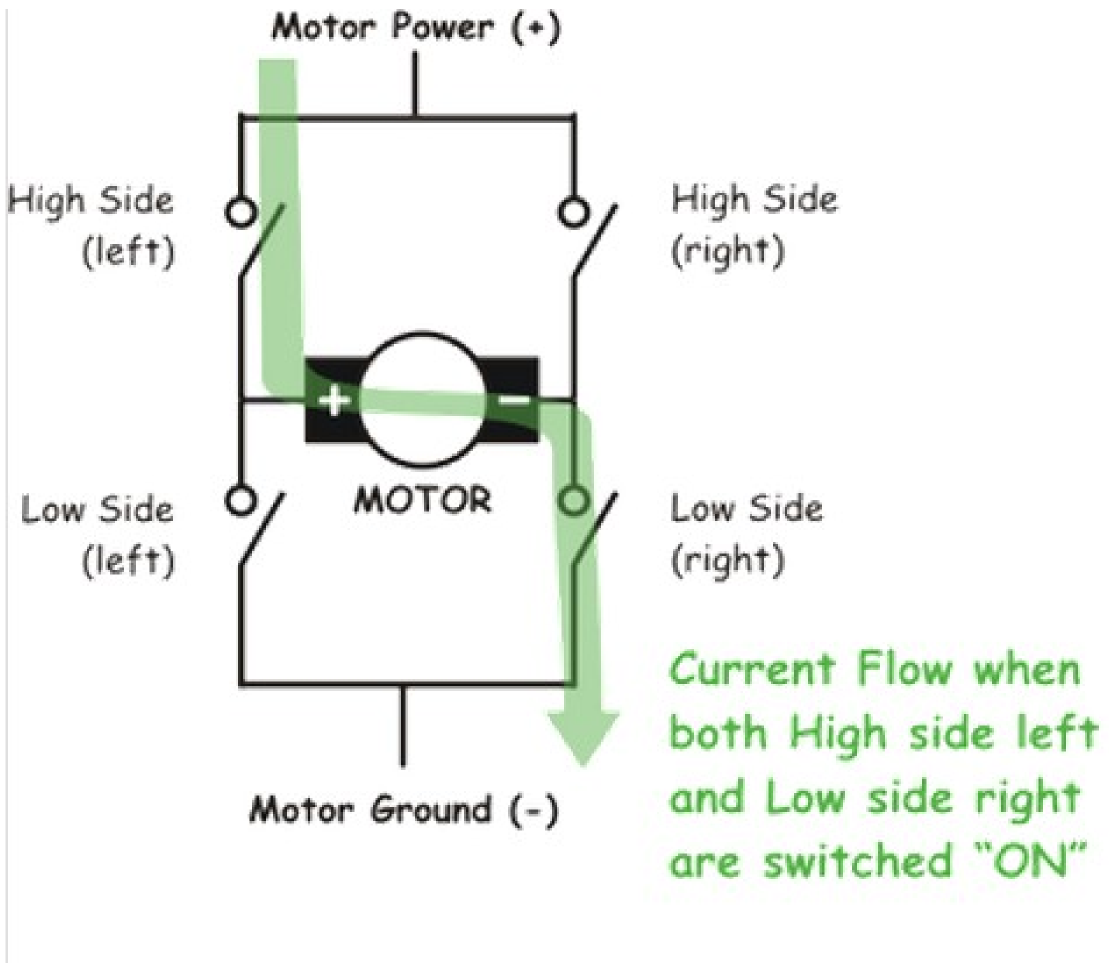


LEDs (PWM)



PWM... Pulse Width Modulation

DC Motors (H - Bridge)



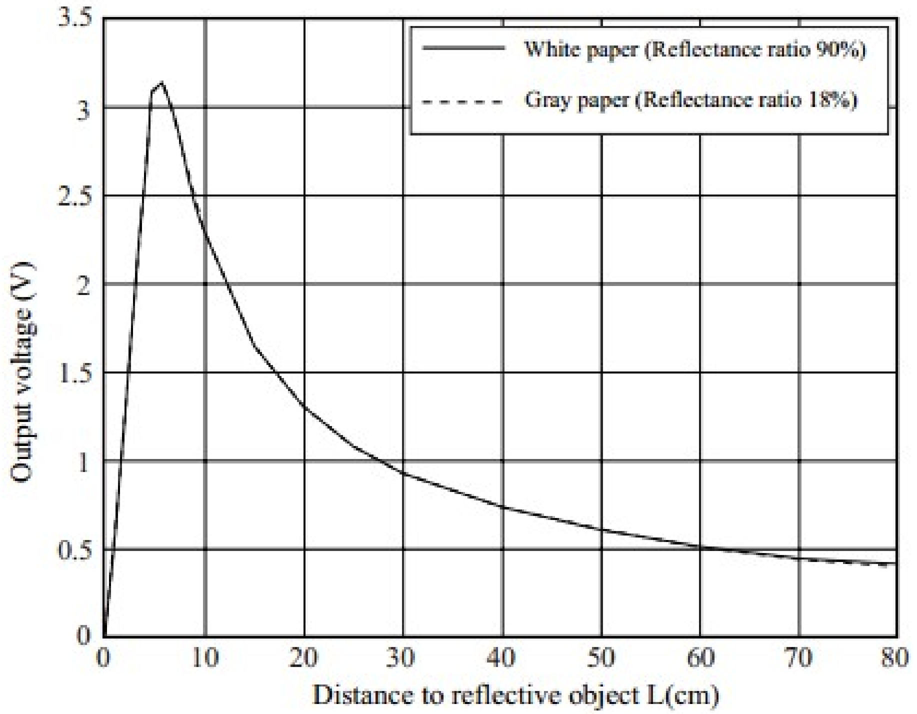
PWM is used to determine velocity

VNH3SP30-E

IR Sensor

GP2Y0A21YK0F

According to Datasheet
from 10 to 80 cm



Sensor Output Voltage will be provided via the Comm Board in hexadecimal notation (0x00 – 0xFF) per sensor.
Example: “sensor: 0x00 0xa2 0xef 0x12 0xf3 0x01 0x00 0x17”

Comm Board Code

File	Content
main.c	entry point + command execution
uart.c	serial communication (low level)
twi.c	i2c communication
timer1.c	PWM servo output
log.h	debug macro
command.c	handle received data (available commands listed here)
robot.c	robot control (high level)