



FTC 2015-2016

Android Based Control System



Agenda

- Control System Overview
- Phone Setup
- Generating the Robot configuration file
- Software Overview
- Autonomous vs Tele-op templates
- Motor/servo control
- Sensor and the Device Interface Module
- Resources

FTC NEW 2015 Control System





Initial Phone Power Up Process

- Remove SIM Card – disables phone activation
- Initial phone configuration – Enables airplane mode, wifi and other phone settings
- Enable USB debugging on robot controller phone
- Install USB drivers on computer
- Connect phone to IDE
- Set phone to charge mode on USB connection



Initial Phone Power Up Process (cont.)

Documentation :

Watch

<https://www.youtube.com/watch?v=n597U6rcl2Y>

and download document

<https://drive.google.com/file/d/0B7uYv9QIObrEcGJnN19BR1Q5cnJ5NWt4MGt2eUhsZi1Uc0tB/view?usp=sharing>



Setup Wireless Debugging

Phil Malon – Youtube Video

<https://www.youtube.com/watch?v=0XZ6EH7BV2M>

Download windows batch files to automate phone configuration.

<http://bit.ly/1LUuTMc>



Development Environment Setup

- Install latest version of JAVA
- Install latest version of Android Studio or App Inventor
- Verify you can connect to Robot Controller phone from development environment.
- Build and install a sample program



Development Environment Setup (cont.)

From the FIRST Website download and read:

- “FTC Training Manual - JAVA Programming for the Next Gen Controller”
- “Next Gen Platform: Team & Mentor Guide”
- “FTC Training Manual - ZTE Channel Changing App”

Links to the above are located at:

<http://www.usfirst.org/roboticsprograms/ftc/team-resources>



Required Google App Store Applications

- From the Google App Store
 - Onto your Driver Station Phone, install the FTC Driver Station Application
 - Onto your Robot Controller Phone, install the FTC Robot Controller Application
 - Onto both phones, install the FTC ZTE Channel Change Application



Robot Configuration Setup

- The robot configuration file tells the software which controllers and sensors are connected to the Robot Controller Application. Each controller and sensor is given a unique name to identify it in the software (i.e. touchsensor1)
- On your robot controller phone run the Robot Controller Application.
- Go to Settings -> Configure Robot

Click New, the application will search for all attached USB devices and display entries for each device. You should rename each device (lower case) and use those names in your code.

Make the device names meaningful, for example if you click on a motor controller entry, you can rename the controller, and name each motor controller port. For example

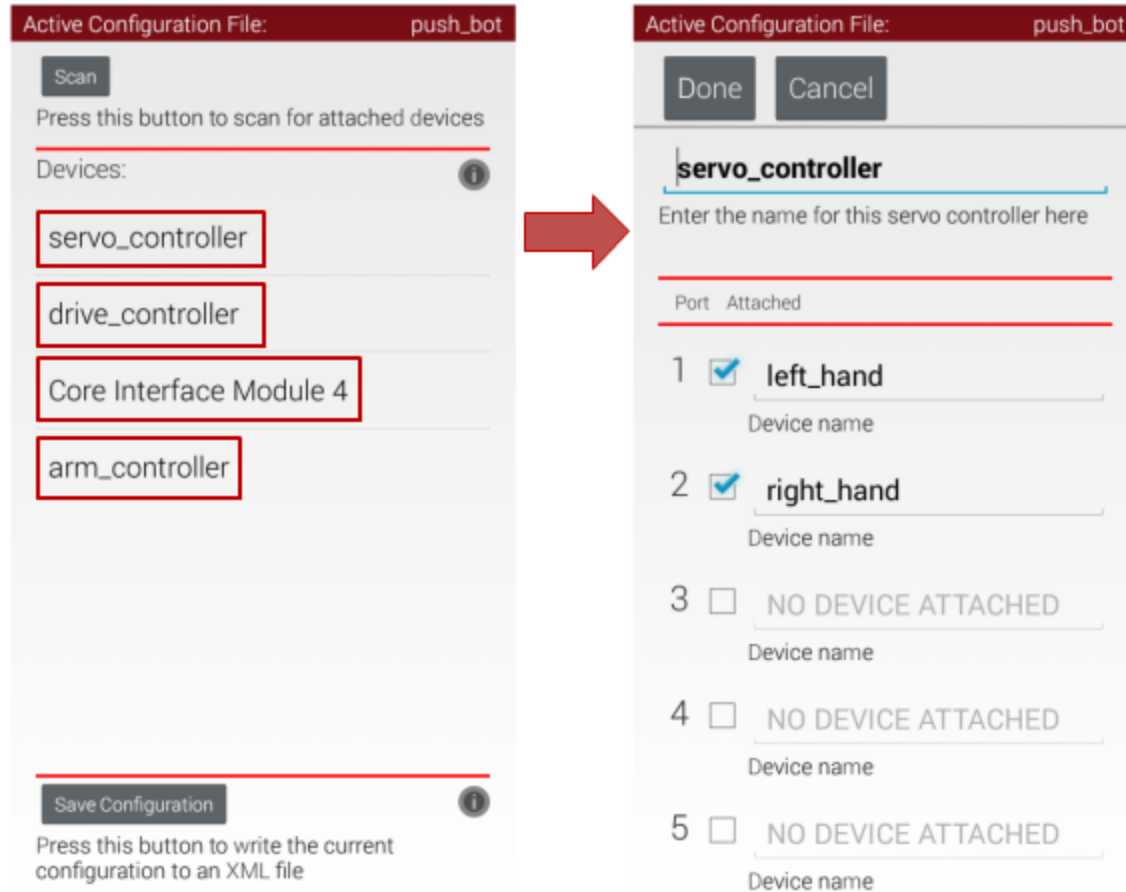
mc_left for left motor controller

left_front for port 1

left_back for port 2

Demonstration...

Robot Configuration Setup (cont.)



The diagram illustrates the process of configuring a robot's servo controllers. It starts with a 'Scan' screen where devices are listed, and then moves to a 'servo_controller' screen where specific devices are selected and named.

Active Configuration File: push_bot

Scan
Press this button to scan for attached devices

Devices:

- servo_controller
- drive_controller
- Core Interface Module 4
- arm_controller

Save Configuration
Press this button to write the current configuration to an XML file

➔

Active Configuration File: push_bot

Done **Cancel**

servo_controller
Enter the name for this servo controller here

Port	Attached	Device name
1	<input checked="" type="checkbox"/>	left_hand
2	<input checked="" type="checkbox"/>	right_hand
3	<input type="checkbox"/>	NO DEVICE ATTACHED
4	<input type="checkbox"/>	NO DEVICE ATTACHED
5	<input type="checkbox"/>	NO DEVICE ATTACHED



Robot H/W Configuration Setup

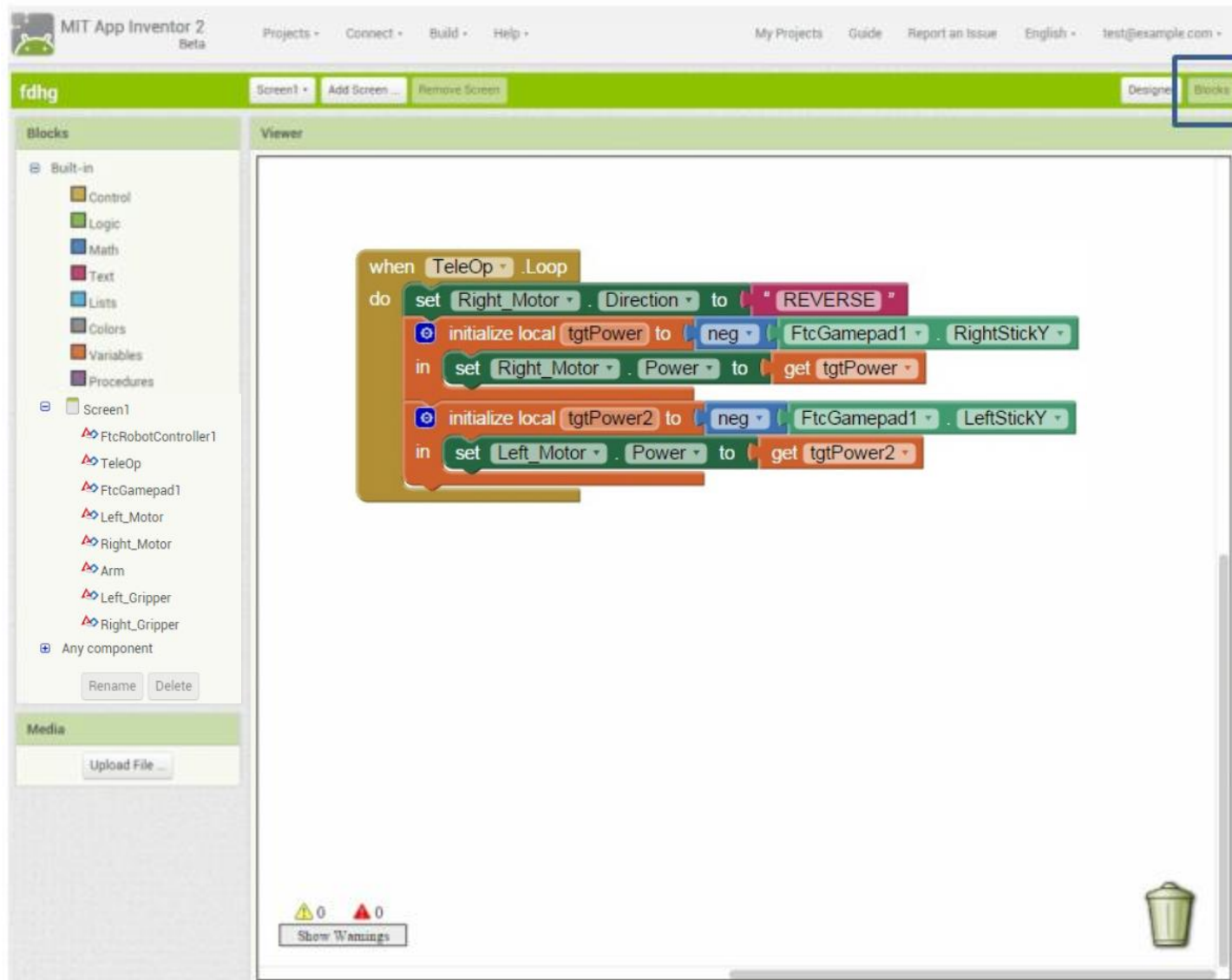
- If you change your robot configuration you have to update the robot hardware mapping, and you may have to update your code.
- Be careful with USB cables if you plug a controller into a different port, you may have to update your robot hardware mapping.
- **BEST PRACTICE –**
 - BUILD A DIAGRAM OF YOUR ROBOT CONFIGURATION (Use it in your engineering notebook)
 - LABEL BOTH ENDS OF YOUR USB CABLES
 - LABEL SENSOR CABLES
 - LABEL MOTOR AND SERVO PORT CABLES



Development Environments

- App Inventor - Is a visual design tool that lets you create Android apps very quickly and intuitively
- Android Studio - Is a full-fledged commercial Java software development environment
- Both use a special SDK provided by Qualcomm with Java classes to access motor, servo, and sensor data.

App Inventor Screen Shot



App Inventor Installing

Installation is fairly simple:

- **Install Google Chrome web browser (free)**
- **Install Oracle VirtualBox Virtual Machine Software (free)**
- **Install the pre-configured App Inventor Server virtual machine image (free) into VirtualBox**
- **Start the App Inventor Server virtual machine.**

Installation manual and the software can be found on Google Drive:

<https://drive.google.com/open?id=0B0...GRHUDE5ZWFORFE>



App Inventor

Intelitek has a nice interactive tutorial on how to perform the install and use the application. It is the same information as in the manual, but leads you through the process:

<http://ftc.edu.intelitek.com/course/view.php?id=7>

•App Inventor Training:

https://drive.google.com/folderview?id=0B0z7bZfPuXgQfIZJUXpmczU4VTVUWkp5eWREUk1EUVdTVWdzRG5BZXh4UGRHUDE5ZWFORFE&usp=drive_web

If you need help:

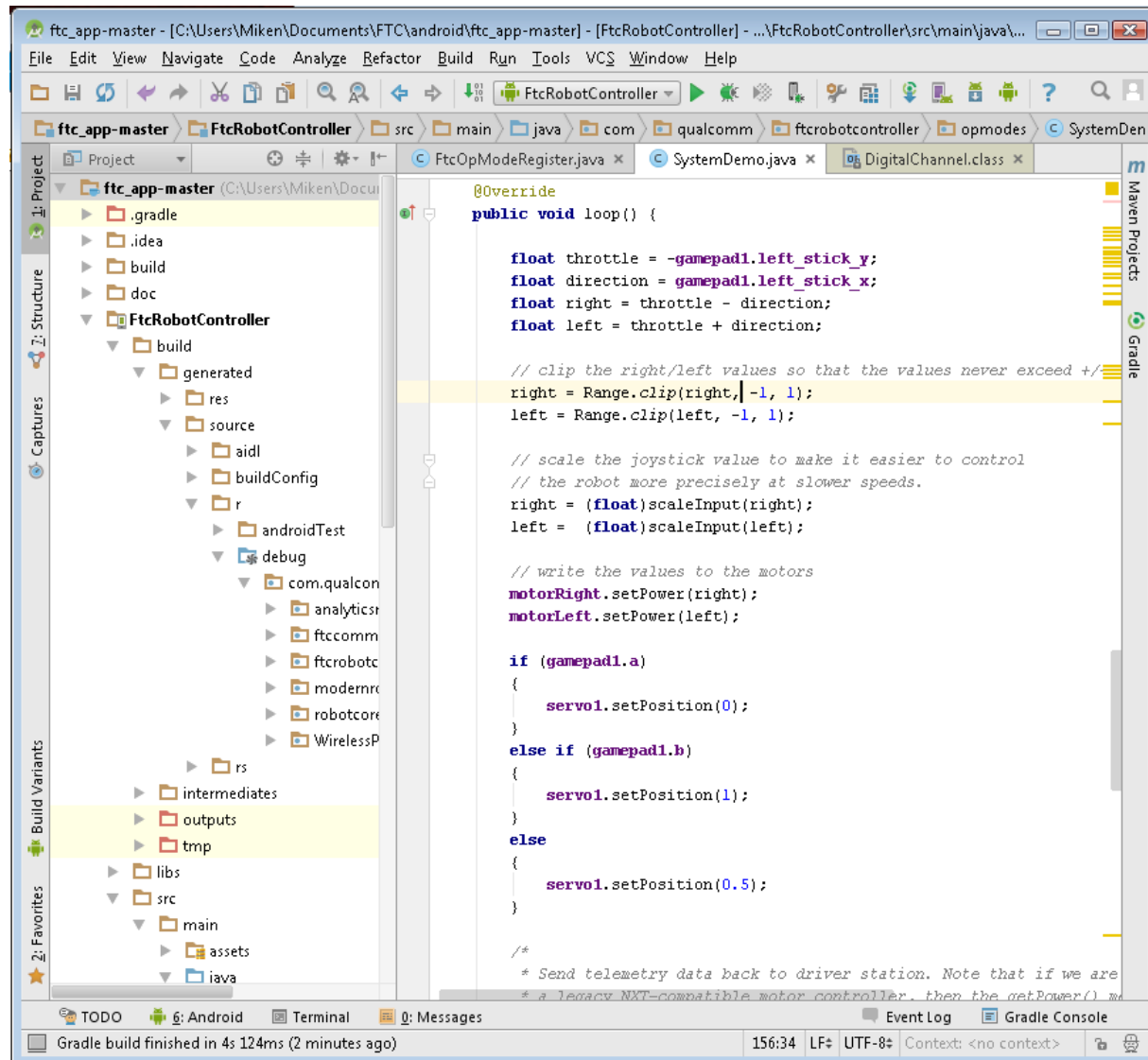
•FTC Technology Forum:

<http://ftcforum.usfirst.org/forumdisplay.php?156-FTC-Technology>

•App Inventor Sub-Forum:

<http://ftcforum.usfirst.org/forumdisplay.php?160-MIT-App-Inventor>

Android Studio Screen Shot





Training Materials

- Intelitek Training for Android Studio and App Inventor:
<http://first.intelitek.com>
- Android Studio GitHub:
https://github.com/ftctechnh/ftc_app
- Android Studio GitHub Training Documents:
https://github.com/ftctechnh/ftc_app/tree/master/doc/tutorial



Programming

In Android Studio the program files that you will need to edit:

- FtcOpModeRegister - Registers opmodes and handles opmodes events
- OpModes (TeleOp) / LinearOpModes (Autonomous)- individual classes (routines) to run robot

TEAMS will inherit from the OpMode or LinearOpmode class to implement autonomous or teleop programs.

Programming

Similarities

- There are many similarities between the C and Java programming languages.
- The core program flow is very similar, with sequences of statements ended with semicolons.
- Many primitive variables are defined similarly, using int, float, double, etc.
- Many program flow commands are similar or identical, like if, else if, else, while, do, for, etc.
- There are many tutorials and books for JAVA development online.



Programming

Differences

- Object Oriented
- Sensor definitions – RobotC Pragma vs hardware map configuration file



Hardware Map Setup

- In Java, hardware is defined through a hardware map.
- Then, in the OpMode Java code, the program defines class instances based on the hardware map in the init or start methods, for example:

```
// defined at top of specific OpMode or LinearOpMode
```

```
DcMotor leftDrive, rightDrive;
```

```
Servo claw;
```

```
TouchSensor touch;
```

```
...
```

```
// defined in OpMode's init or start method, or LinearOpMode's runOpMode method
```

```
leftDrive = hardwareMap.dcMotor.get("motor_1"); // motor_1 set in Configure Robot
```

```
claw = hardwareMap.servo.get("servo_1"); // servo_1 set in Configure Robot
```

```
touch = hardwareMap.touchSensor.get("touch_sensor_1"); // ...
```



Reading Joystick / Gamepad Settings

- The stick settings range from -1 to 1.
- For example:

```
leftDrive.setPower( gamepad1.left_stick_y); // stick is -1 to 1 !!
if (gamepad1.a) { // uses the names of the buttons rather than button numbers!
    // do stuff here
}
if (gamepad1.dpad_down) { // uses dpad_up, _down, _left, _right
    // do stuff here if bottom top hat is pressed
}
...
```
- **Important change: for the y_sticks, +1 corresponds to pulling the stick down and -1 corresponds to pushing the stick up!!! This is the opposite sign polarity from what is done in RobotC and from the normal cartesian coordinate plane.**
- See the full SDK documentation for the syntax of all gamepad settings.



Initializing at the Start of Operation

Initial settings are set in the init OpMode method. For example:

```
float driveSpeed;  
@Override  
public void init() {  
    // setup claw servo using hardware map - not shown here  
    ...  
    claw.setPosition(0.75);  
    driveSpeed = 0.5;  
    ...  
}
```

One key difference is that the init method writes to hardware only once at the end of the method. Teams that, for example, used the old RobotC initializeRobot function to spend multiple seconds reading from a gyro sensor for initial calibration will NOT be able to do the same function by simply copying that functionality into the init method. **(This may change in a future release of the SDK.)**



Main Program Loop

- In RobotC, repeated operation is implemented in a loop structure after the `waitForStart`. For example, a simple teleop routine structure might be:

```
void main() {  
    waitForStart();  
    while (true) {  
        getJoystickSettings(joystick1);           // Read the joysticks  
        motor[motorleft] = joystick1.y1;          // Set motor power  
        motor[motorright] = joystick1.y2;         // Set motor power  
        wait1Msec(20);  
    }  
}
```

- In Java, the outer structure of the above loop is already provided by the `OpMode` class and underlying processing. The loop method is essentially only the inner loop of the above RobotC example, or the code between the `// start doing stuff` and the `// end doing stuff` comments.

```
@Override  
public void loop() {  
    leftDrive.setPower( gamepad1.left_stick_y);    // Set motor power  
    rightDrive.setPower( gamepad1.right_stick_y);  // stick is -1 to 1 !!  
}
```



Setting Motor Power and Servo Settings

- In RobotC, motor powers and servo settings are set through arrays, for example:

```
motor[leftDrive]=100;    // from -100 to 100  
servo[claw]=128;         // from 0 to 255
```
- In Java, motor powers and servo settings are set using methods of the corresponding classes, for example:

```
leftDrive.setPower(1);    // from -1 to 1 !!  
claw.setPosition(0.5);    // from 0 to 1 !!
```



LinearOpMode (Autonomous)

- In version 1.05 of the FTC SDK, a new class called LinearOpMode was introduced to allow a coding style similar to the type of RobotC code shown above. In a LinearOpMode, we only override one new method: runOpMode(). We don't have to worry about the init, start, loop, or stop methods.
- The LinearOpMode waitForStart(), sleep() and waitOneHardwareCycle() methods allow for motor and servo values to be written and sensor and encoder values to be read before the program continues.// setup the motors using hardware map calls here, as in a normal OpMode



LinearOpMode (Autonomous)

- In RobotC, a simple autonomous routine's core code might look like this:

```
waitForStart();
```

```
motor[leftDrive]=100; // drive forward
```

```
motor[rightDrive]=100;
```

```
wait1Msec(2000);           // or wait for a motor encoder to reach a desired setting
```

```
motor[leftDrive]=0;        // stop
```

```
motor[rightDrive]=0;
```

```
....
```

- To mimic the above RobotC code using the LinearOpMode method is shown below.

```
// setup the motors using hardware map calls here, as in a normal OpMode
```

```
.....
```

```
waitForStart();           // new method only in LinearOpMode!
```

```
// this pauses the program until the start button is pressed
```

```
// on the Driver Station App
```

```
leftDrive.setPower(1);    // drive forward
```

```
rightDrive.setPower(1);
```

```
sleep(2000);              // or wait for a motor encoder to reach a desired setting
```

```
leftDrive.setPower(0);    // stop
```

```
rightDrive.setPower(0);
```

```
waitOneHardwareCycle(); // push the stop out to the motors.
```



What we didn't talk about

- **MultiThreading**
- **Motor/Servo Update Rates**
- **Sensor Update Rates**
- **Additional information on Start/Init/Loop/Stop functions**
- **And a whole lot more ...**

More information can be found at:

From RobotC to Java for FTC Programmers - *William Gardner* –

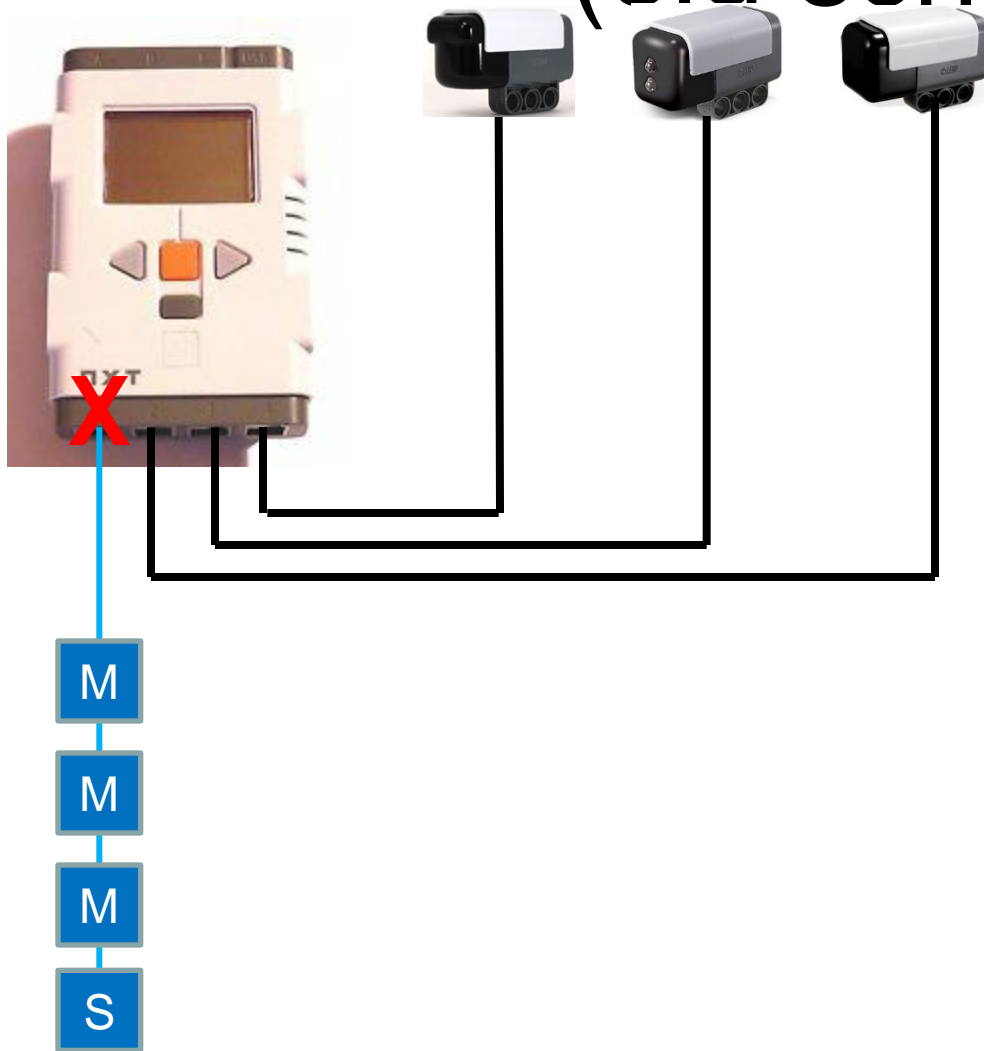
<http://cheer4ftc.blogspot.com/p/2015-technology.html>



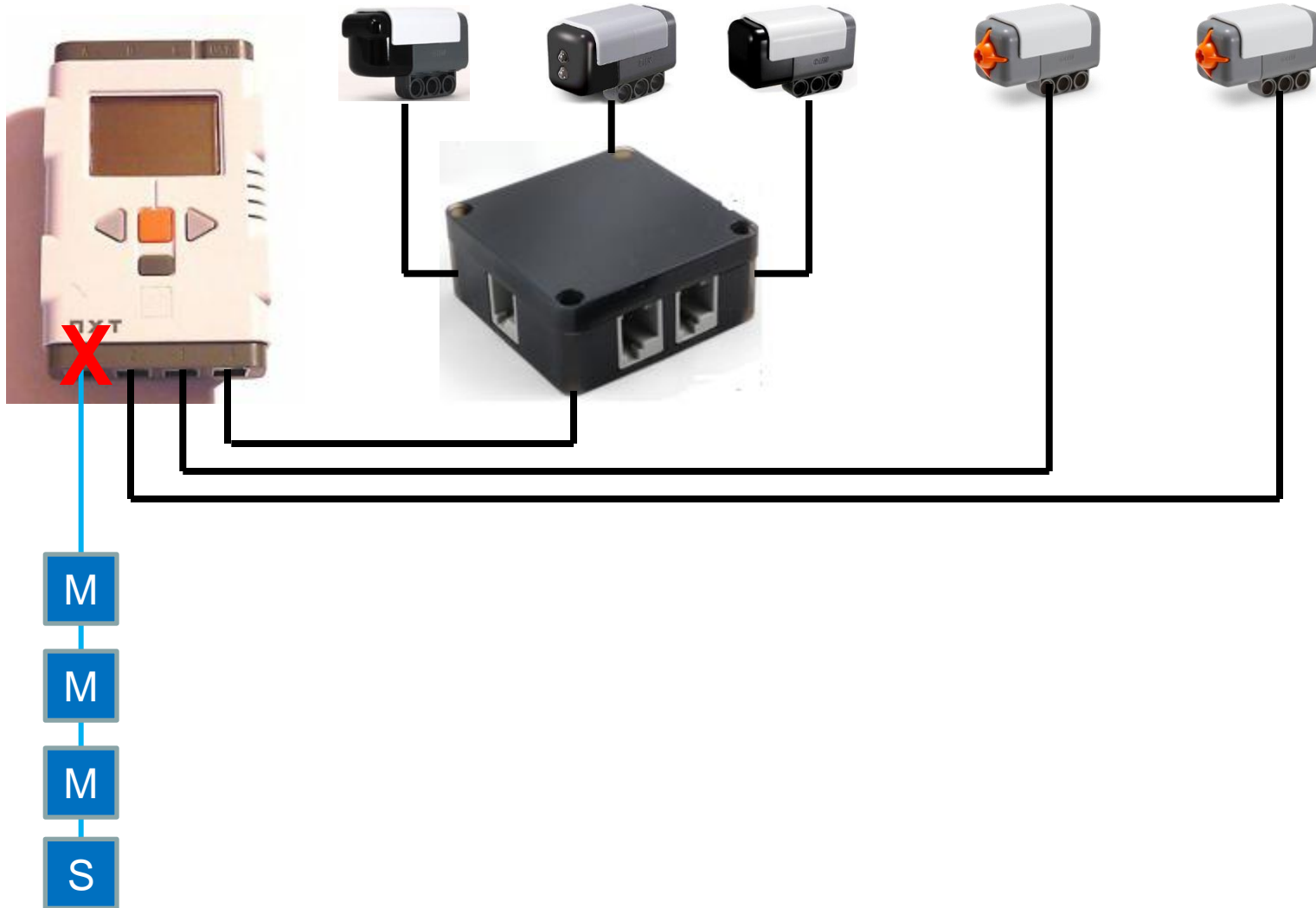
Getting Sensored

**Using existing and custom sensor
with the New
Device Interface Module**

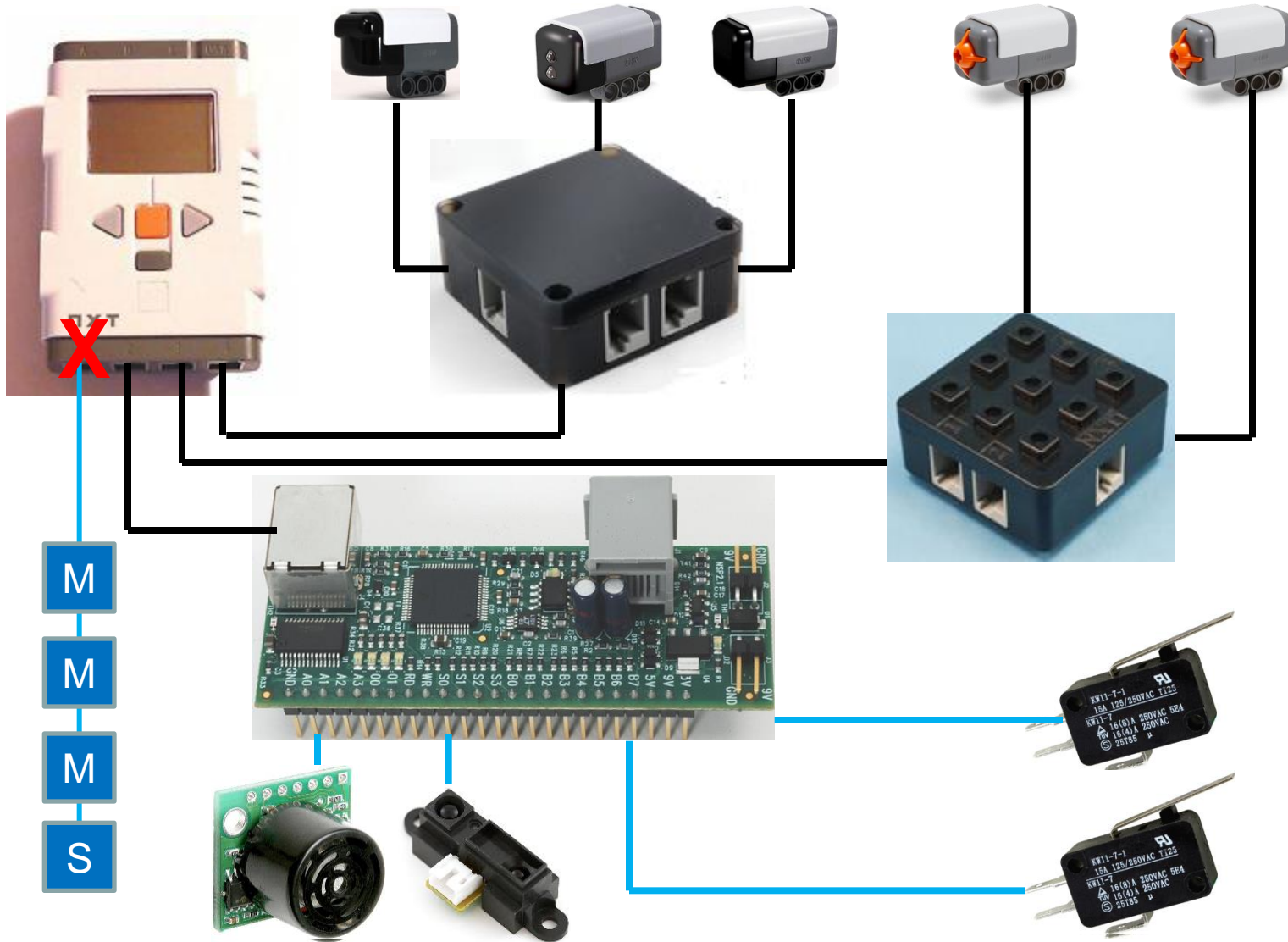
NXT Sensor Interface (old school)



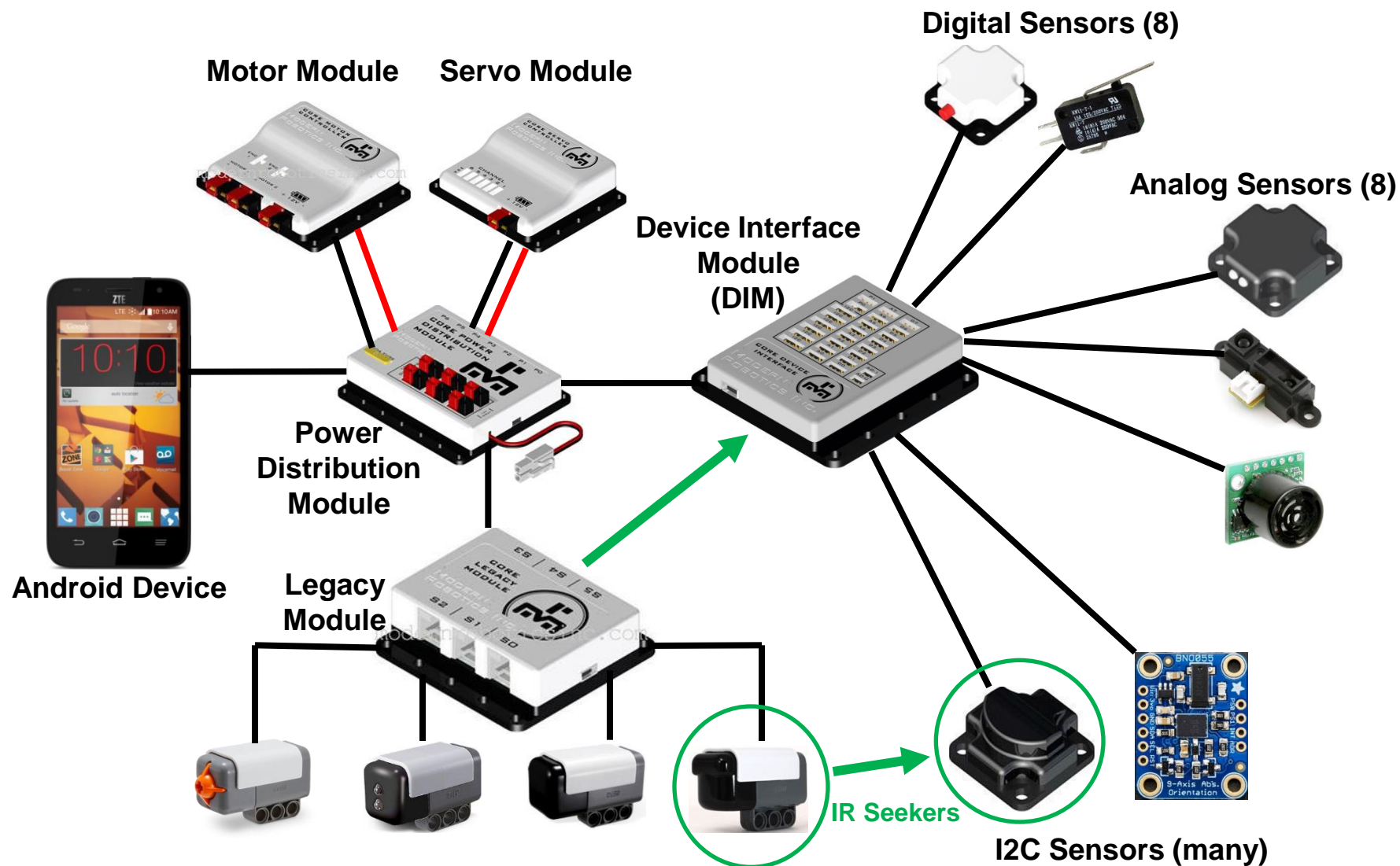
NXT Sensor Interface



NXT Sensor Interface



Android Sensor Interfaces



Device Interface Module

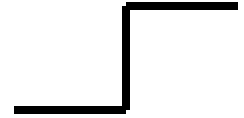
- This module hosts all independent sensors.
- It allows you to interface to the following types of sensors:
 - Digital (in/out)
 - Analog (in/out)
 - PWM (out)
 - I2C
- This modules opens up a lot of capability for you to take advantage of!!!!!!



Logic Levels

- **The Device Interface module is a 5 volt (V) device. This means it can safely operation in the range of 0-5V. This is a consideration when selecting items to interface to the Device Interface Module.**
- **A logic 0 results when the voltage at the input/output pin is “near” ground (0-1V). A logic 1 results when the voltage at the input/output pin is “near” 5V (4V-5V). Anything Voltage applied to a digital pin outside of the 0 or 1 voltage range is indeterminate (could read a 0 or 1).**
- **Many sensors and other devices are now being designed to operate at lower voltages (i.e. 3.3V, 2.0V, etc.). It is still possible to use these lower voltage devices, but you may need to make allowances in your code or add in some extra interface electronics or you risk damaging the part.**
- **Keep in mind that the Device Interface Module can only source 150 milliamps of current. That is not a lot, so make sure you add up all your loads so you don’t go over.**

Digital (In/Out)

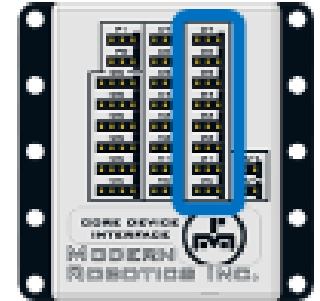
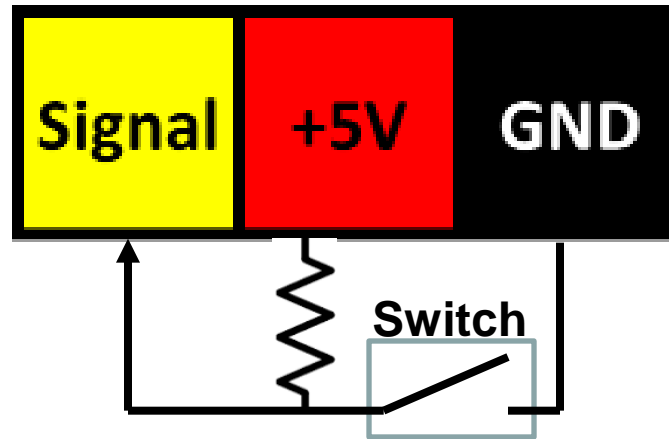


- **Digital signals can only represent one of two states at a given moment: a logic 0 (low, off, ground) OR a logic 1 (high, on, voltage),**
- **There are 8 digital ports on the modules with each port being configurable as an input or an output.**
- **The port and its associated sensor should be aligned in direction (one is an input, one is an output).**
- **The software will read a 0 or 1 depending on the voltage level on the pin (0 or 5V).**
- **If the port is an input, it must be driven high or low and not be allowed to “float”. A pull-up resistor may need to be added in some cases.**
- **This is a very simple interface to use if you only need two states.**

Digital Sensors

Digital Sensor Examples:

- Toggle/slide switches
- Limit switches
- Magnetic switches
- Optical switches
- Interrupts signals
- Momentary switch (touch sensor)



Toggle Switch



Limit Switch



Magnetic Switch



Momentary Switch



Optical Switch



```
public class digitalDemo extends OpMode          // Class begins.  
{  
    DigitalChannel digital;                     // object variable.  
  
    public void init()                          // Automatically called once at program start.  
    {  
        digital = hardwareMap.digitalChannel.get("touch1"); // Create the analog object  
    }  
  
    public void start()                         // Called once at start of teleop.  
    {                                           // Nothing to do for now.  
    }  
  
    public void loop()                         // Automatically called repeatedly during teleop  
    {  
        boolean digVal = digital.getState();   // Read pin into variable.  
  
        telemetry.addData("Digital1:", String.format("%1d", (digVal ? 1 : 0))); // print to screen  
    }  
  
    public void stop()                        // Automatically called at end of teleop.  
    {                                         // Nothing to do for now.  
    }  
}
```

Analog (In)



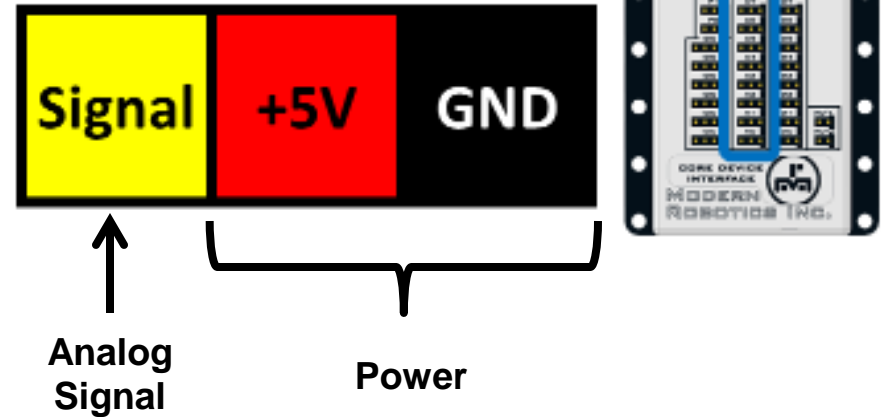
- **Analog signals can have a voltage anywhere in a given range and are used to convey any value within the range.**
- **Analog Signal (definition): Is dependent on the sensor which is creating the analog signal.**
- **The analog ports on the Device Interface Module operation between 0 and 5V. Thus, any voltage between those two values are acceptable.**
- **In order for the processor to work with the analog signal, it is first turned into a 10-bit number (0-1023). The lower the voltage, the lower the number (0V=0), the higher the voltage, the higher the number (5V=1024), and anything in between is proportional (i.e. 2.5V=512).**
- **Since the sensor must generate the voltage, there is rarely a need for any type of pull-up resistor.**

Analog Sensors

Examples:

- Ultrasonic range finders
- IR range finders
- Gyros
- Accelerometers

Code Example:



IR Proximity
Sensor



Ultrasonic
Proximity Sensor



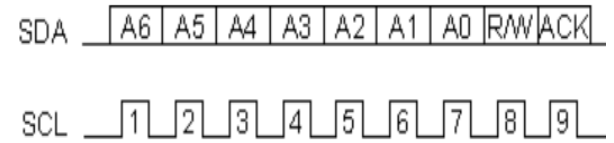
IR Proximity
Sensor



Accelerometer

[illegible]

I²C (Bus)



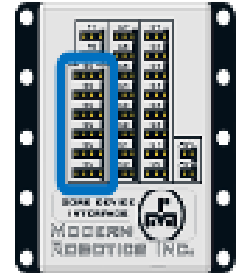
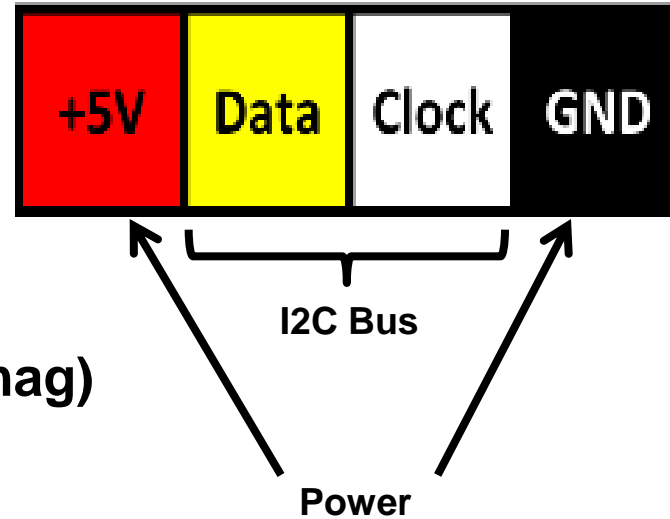
- The I²C interface allows “smart devices” to transfer digital data (numbers) both to and from the sensor.
- These digital transfers allow for more accurate information to be exchanged.
- This allows for greater control of the sensor.
- This enhanced communications allows for more sensors to be put into a single device with each sensor being accessed separately (gyros, accelerometers, and compasses).
- This is a huge enabler for the use of better, faster, smarter, and cheaper sensors on future FTC robots.

I²C Sensors

Examples:

- IR Seekers
- GPS
- Color sensors
- N DoF sensors (gyro/accel/mag)
- Orientation Processors

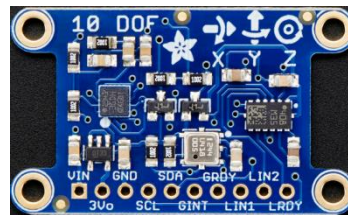
Code Example:



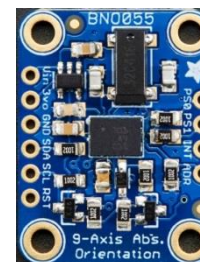
IR Seeker V3



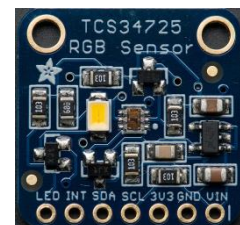
Inertial
Measurement Unit



Orientation
Processor



RGB Color Sensor

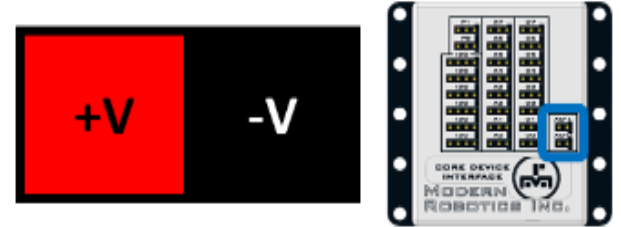




Coding an I2C Port

- Add code here.

Analog Output



- The analog output ports can generate specific analog patterns (sine waves, saw tooth, etc.) and at specified frequencies.
- The analog output ports on the DIM can generate signals of -4V to 4V.
- There is limited information on this port and limited used for the functionality it provides, thus it will not be discussed further at in this presentation.

PWM (out)

- The digital signal generated by a PWM ports is similar to those generated by the Servo Module. They both cases, it is a repeating square wave pulse with a specific duty cycle (ratio of high to low).



- Servo signals must conform to very specific timing restrictions, where as the PWM ports are not constricted by those same restrictions.
- The PWM ports output a digital signal of 0 or 5V with a resolution of 1uS and a frequency of 65 mSec.
- There is limited information on this port and limited used for the functionality it provides, therefore it will not be discussed further in this presentation.

The Sensor Sandbox



- The Arduino board is a great way to work with sensors without tying up your robot hardware.
- Arduinos have many of the same interfaces the Device Interface module (digital, analog, and i2c).
- Arduinos boards are cheap (\$10-\$20), software is free, and they are easy to use.
- This will allow a team to try many different sensors, then use the ones they like best.

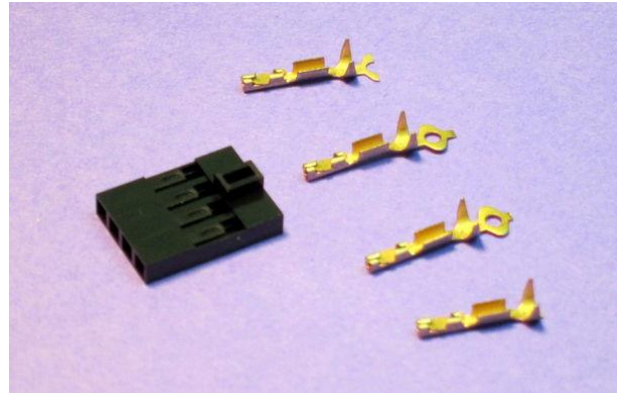
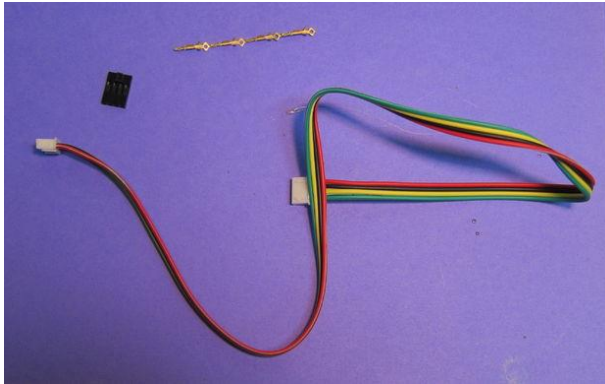


A Few Cautions

With more flexibility comes more responsibility:

- **Be aware of the voltage/logic levels when selecting sensors. 5 volts can damage a 3.3 volt sensor.**
- **Make sure you configure the direction of any digital pins you use and hook sensor outputs to digital inputs. Hooking outputs to outputs could damage the module, sensor, or both!!!**
- **Be aware of the total power your devices draw from the Device Interface module. It has a limit of 150mA total.**
- **Don't swap around your sensor cables on the Device Interface Module. Your software will read whatever is connected to the port, even if it is the wrong sensor.**
- **Don't plug your sensor cables in backwards!!!**

BEWARE AndyMark Encoder Cable



- AndyMark AM-2965 cables
These are the double ended cables where both ends have the AndyMark motor connector on them. You can make two motor encoder cables from each one of these.
- AndyMark AM-2992 cable
BEWARE 4 Pin housing not keyed for FTC motor controllers. Make sure BLACK cable is towards bottom of motor controller, upgrade to KEYED housing
- Digi-Key Conn Housing 609-2396-ND
- Digi-Key Mini-PV Crimp Wire rcpt 609-3620-1-ND



Recommendations

- **Buy a few Arduino for doing sensor prototyping.**
- **Buy some USB-Male A to 90 degree USB mini adapters or cables. This will save space and your connections.**
- **Buy APP crimper/connectors. Replace all Tamayo (white) connectors.**
- **Sensor pig tail kit available from Modern Robotics Inc (MRI).**

APP Connectors



Recommendation to Leagues:

- Buy an Anderson Crimper and medium supply of connectors (couple hundred). This can be used to support teams that cannot afford to buy their own.

Recommendation to teams:

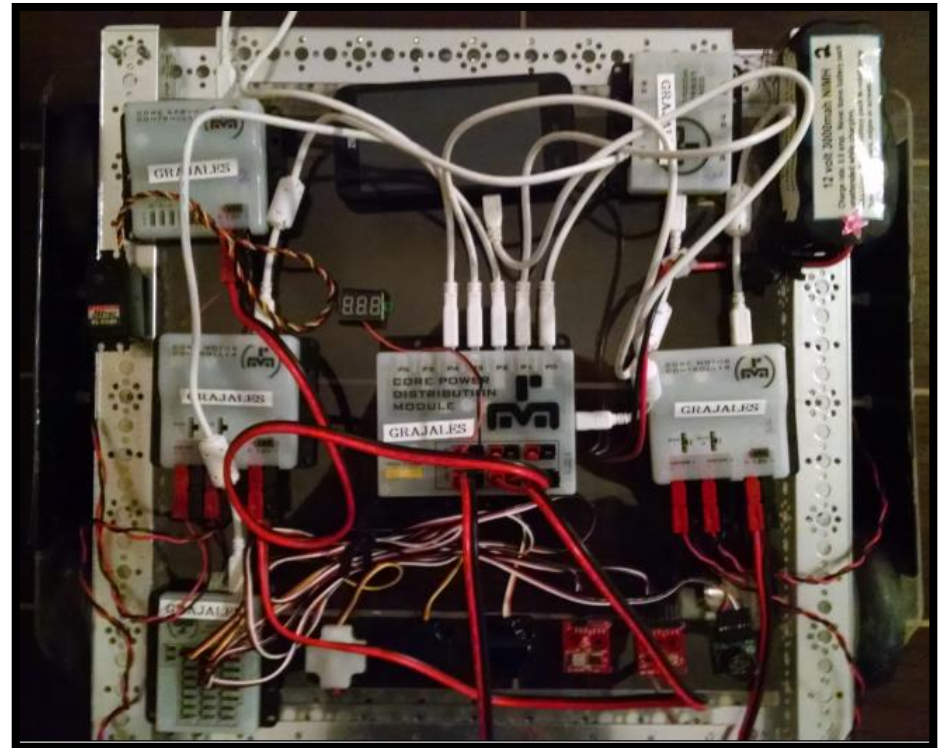
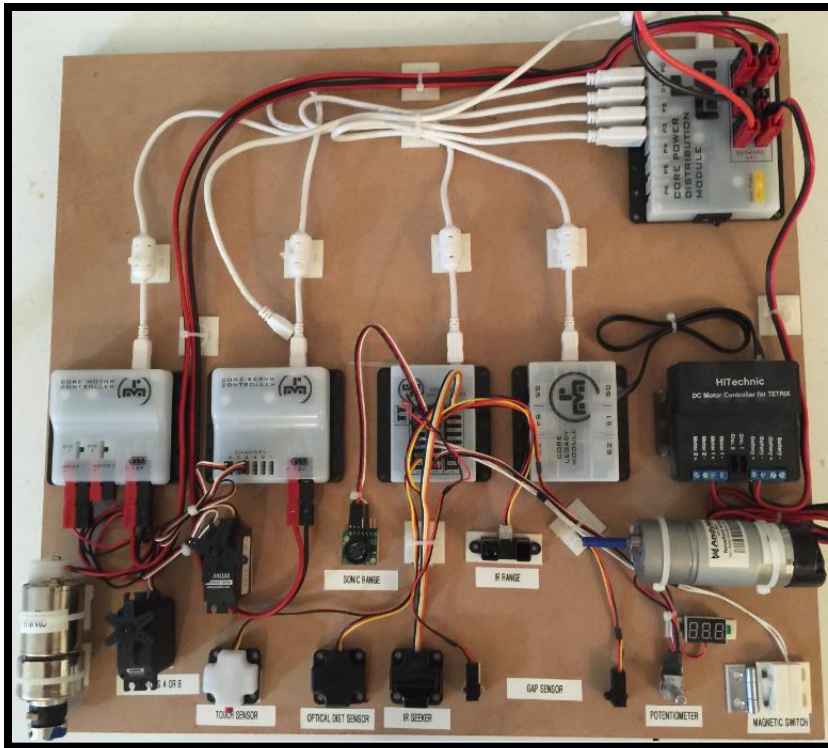
- Buy an Anderson Crimper and a small supply of connectors. You will need to add connectors to motor wires and to make any custom length cables you wish.

APP Connectors (cont.)

APP parts available from <http://www.powerwerx.com>

- **Crimpers**: These crimpers are made specifically for APP terminals and make the job much easier.
- **WP15** (15 Amp) Connectors: Need 1 of these connectors for each battery charger you decided to convert, and one connector for each motor. **Recommend** teams get the set of 25 (WP15-25)
- **WP30** (30 amp) Connectors: Need one of these for each battery you convert, and one for the Power Distribution Module on the robot. **Recommend** teams get the set of 25 (WP30-25).
- **PP15**: Extra 15 amp terminals. Allows for bad crimps and reuse of the connector bodies. **Recommend** teams get 30.
- **PP30**: Extra 30 amp crimp on terminals. Allows for bad crimps and reuse of the connector bodies. **Recommend** teams get 20.

Lunchtime Demos





Resources

- www.modernroboticsinc.com – Android based control system
- www.usfirst.org – National FTC website
- www.flfirst.org – Florida FTC website
- www.hitechnix.com – Legacy Electronics supplies
- www.powerwerx.com – Anderson power poll connectors
- www.tetrix.com – FTC robot hardware
- www.adafruit.com – Sensors and electronics
- www.sparkfun.com – Sensors and electronics
- www.pololu.com - Sensors and electronics



Thanks

- **From RobotC to Java for FTC Programmers - *William Gardner* - <http://cheer4ftc.blogspot.com/p/2015-technology.html>**
- **Phil Malone (Mentor FTC Team 2818 G-Force)**
- **Modern robotics**
- **Adafruit Electronics**
- **Spark Fun Electronics.**

Questions

**Ask now
or
ask at lunch
or
ask later on**



But please ask!!!