m-Block

By Wilmer Arellano

- With the understanding that:
  - Public Domain — Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.
  - Other Rights — In no way are any of the following rights affected by the license:
    - Your fair dealing or fair use rights, or other applicable copyright exceptions and limitations;
    - The author's moral rights;
    - Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.
  - Notice — For any reuse or distribution, you must make clear to others the license terms of this work.

mBot

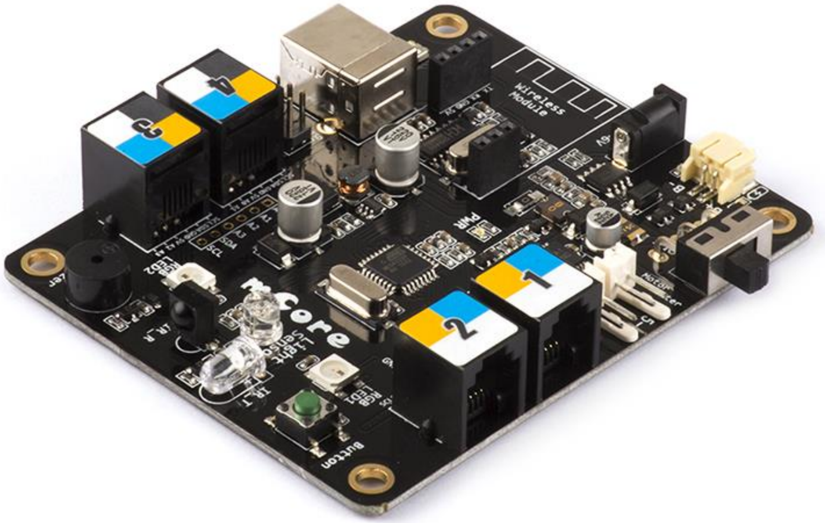
- mBot is an all-in-one solution for kids to enjoy the hands-on experience about programming, electronics, and robotics. Working with mBlock inspired by Scratch 2.0, you can use Bluetooth or 2.4GHz wireless module to connect with mBot (By different version), this easy-to-assemble mBot

mBot

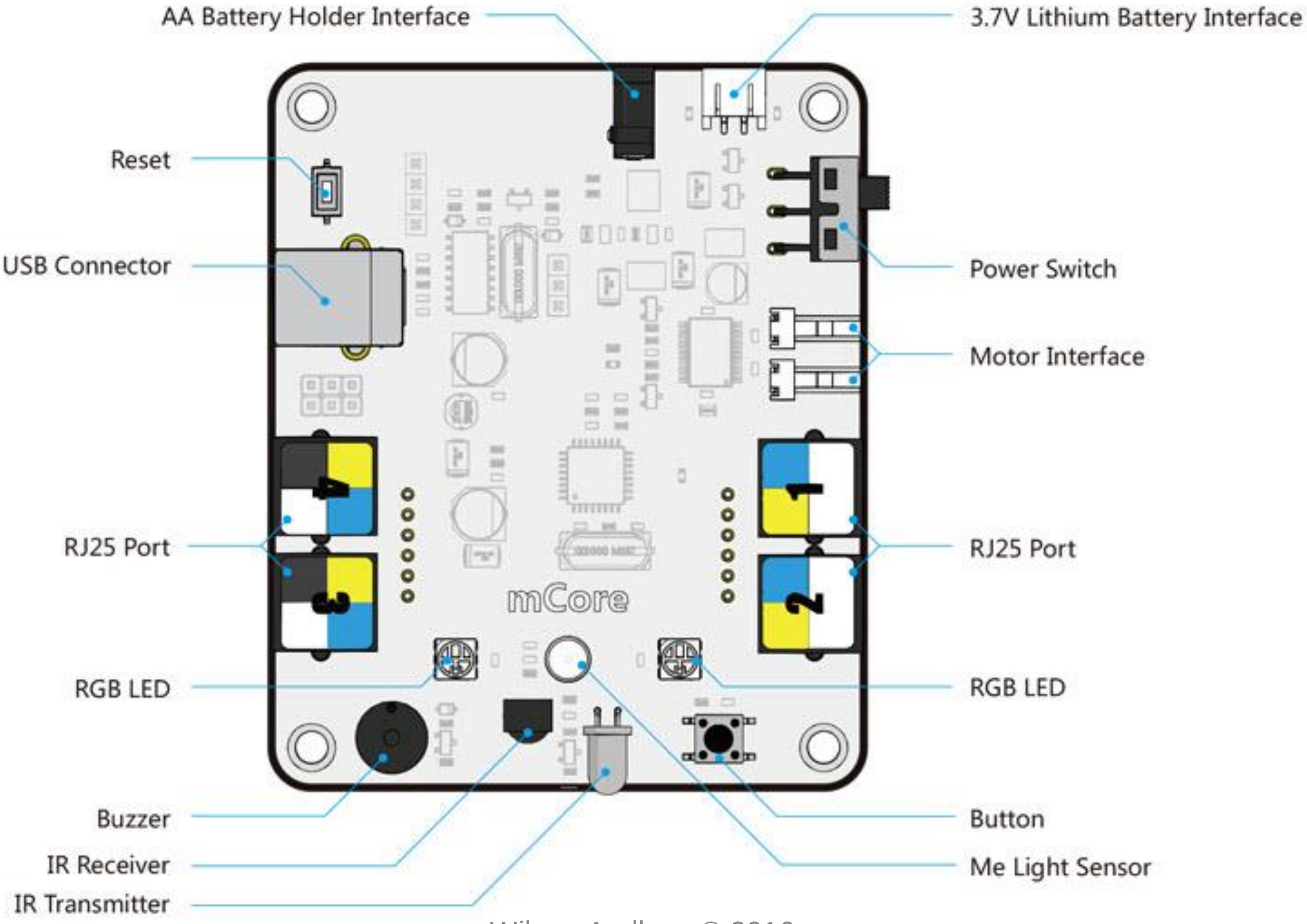
- Treat gently robot is fragile, specially additional modules (line follower, ultrasound, 2.4G, Bluetooth, etc.) and wheels

What is mCore?

- mCore is an easy-to-use main control board specially designed for mBot. Based on Arduino Uno, mCore integrates various onboard sensors, such as buzzer, light sensor, RGB LED, etc., which provides you an easier way to start learning electronics.

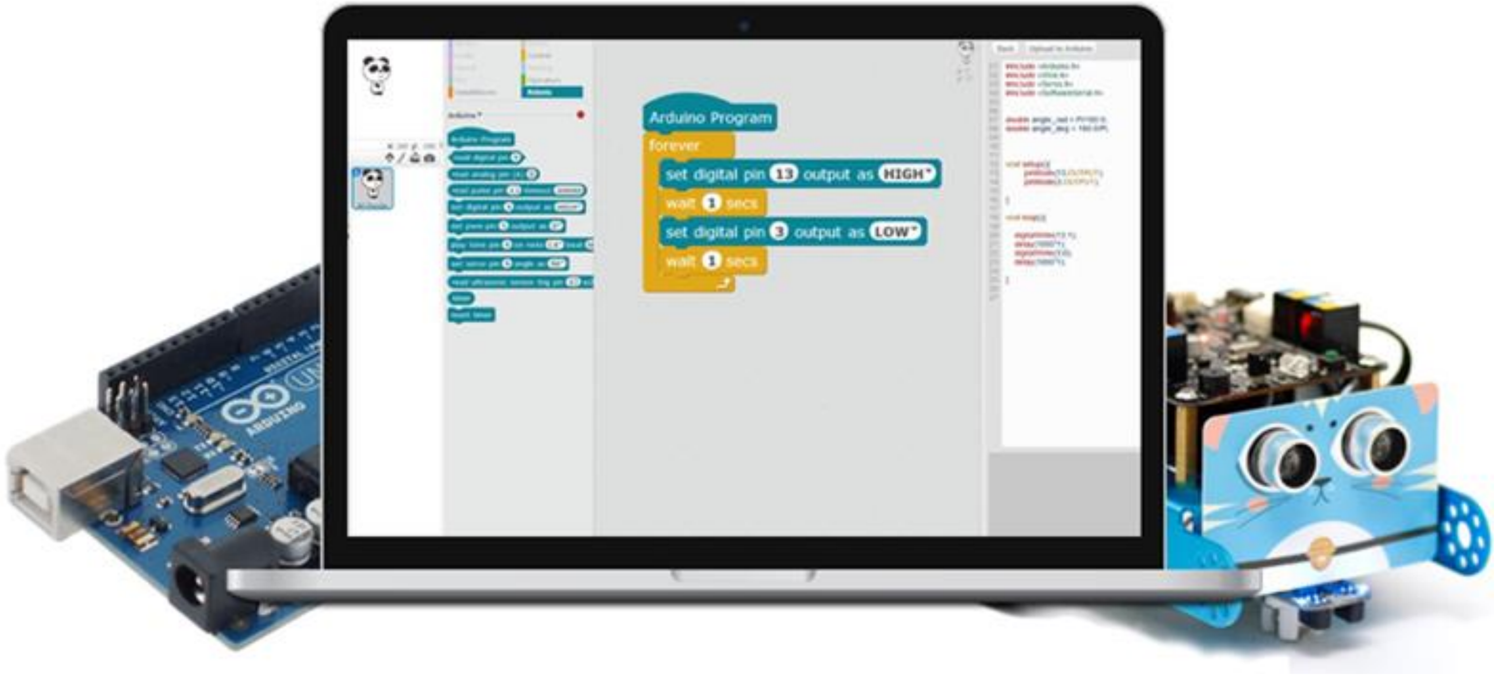


What is mCore?



mBlock

- mBlock is a graphical programming environment based on Scratch 2.0 Open Source Code that makes it easy to program Arduino projects and create interactive applications.



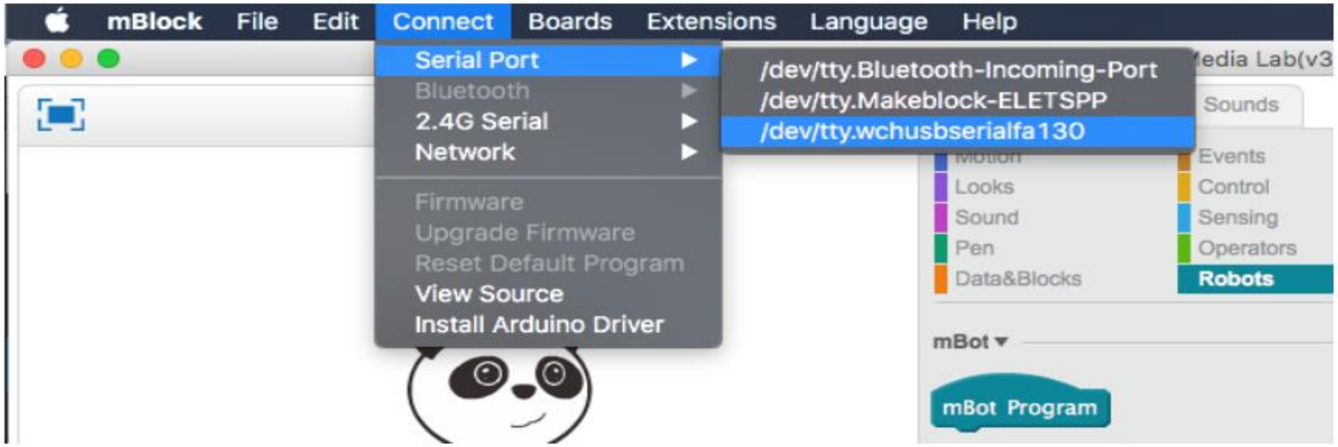
```
Arduino Program
forever
    set digital pin 13 output as HIGH
    wait 1 secs
    set digital pin 3 output as LOW
    wait 1 secs
```

Tutorials

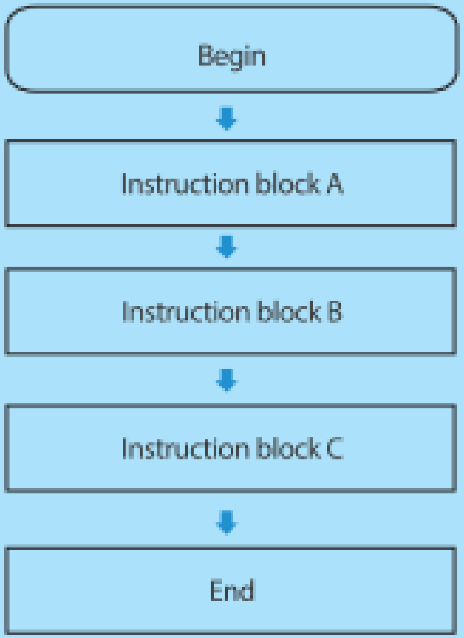
- Resources page
- Scratch Tutorial
- m-Block Tutorial

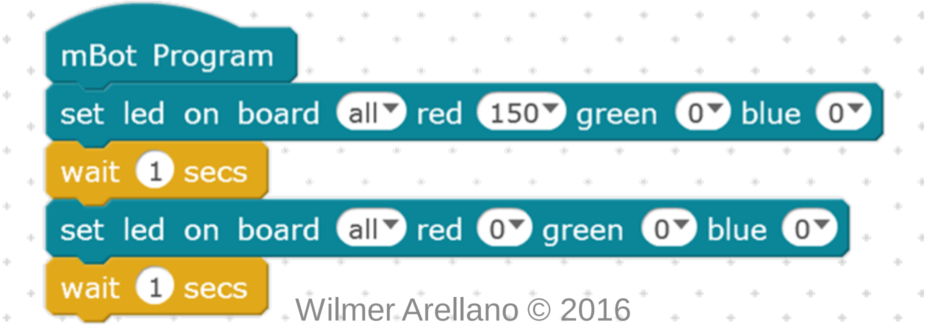
Connect Your Robot Through USB Cables

- 1.use a USB cable to connect between your robot/controller board and your computer
- 2.make sure the robot is powered on (using the power switch)
- 3.select “Connect”, “Serial Port”, then the option corresponding to your robot. If you are using Windows, it should be something like “COM” and a number; if you are using a Mac, it starts like “/dev/tty.wchusbserial”. You may try different options if it does not works.



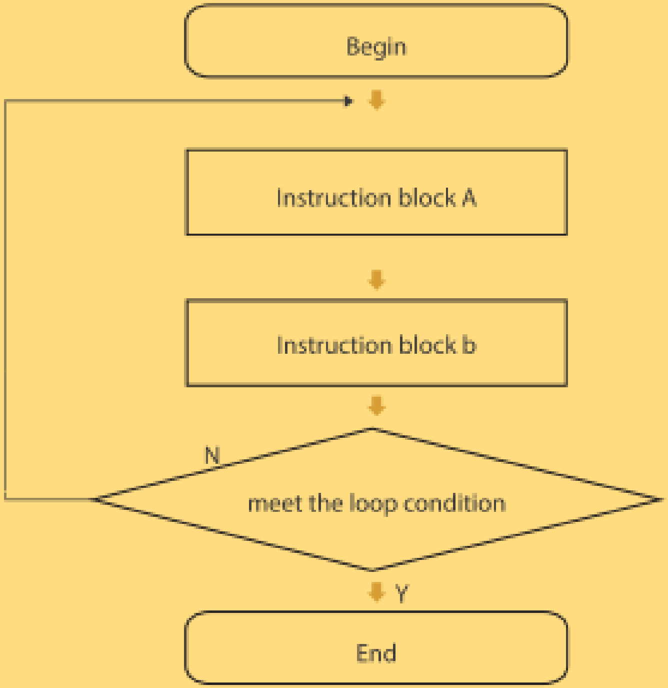
Programming Structure

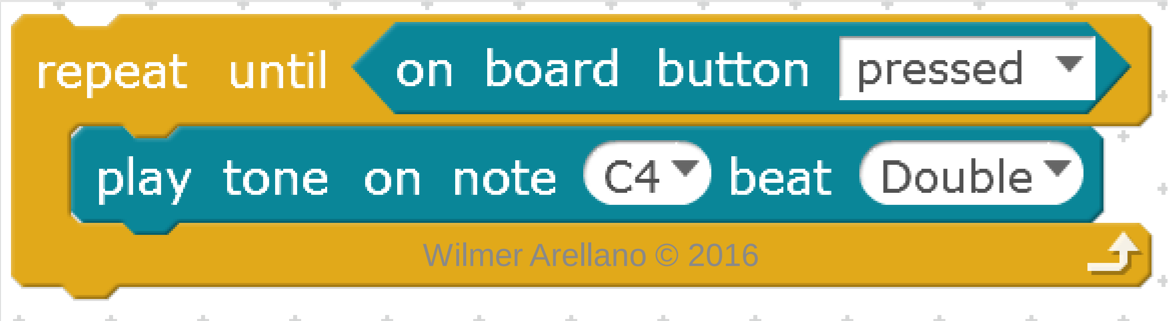
Programming structure	Thinking Process
Sequence structure description: The script begins running from the first block, followed by all the other blocks executing in order. This is the sequential structure. The diagram on the right is a standard sequence structure. After the program starts, it implements three blocks in turn and finally ends. The sequential structure is the basis for the running way of the program.	 Begin Instruction block A Instruction block B Instruction block C End



```
mBot Program
set led on board all▾ red 150▾ green 0▾ blue 0▾
wait 1 secs
set led on board all▾ red 0▾ green 0▾ blue 0▾
wait 1 secs
```

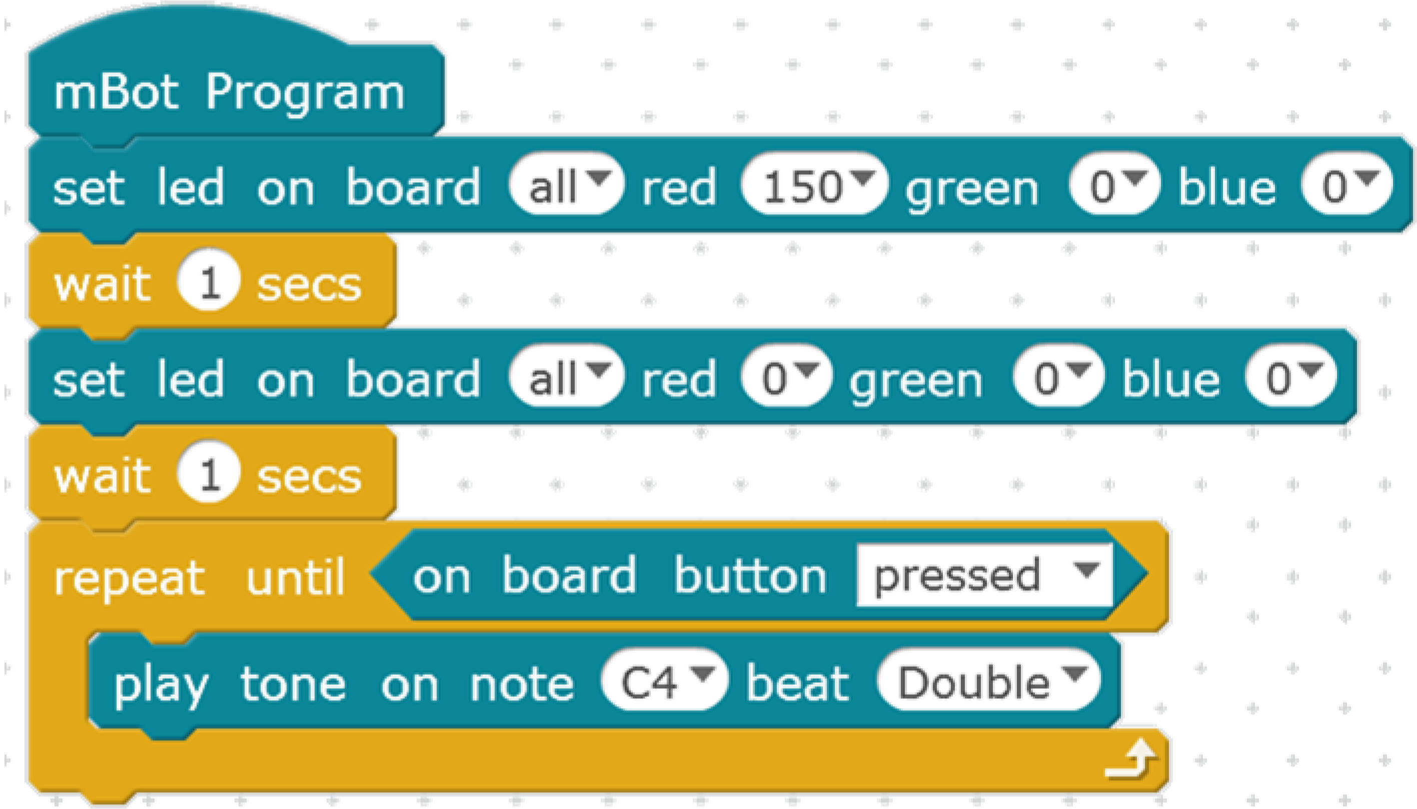
Programming Structure

Programming Structure	Thinking Process
Loop Structure (Cycle Structure): Loop structure is the structure which repeats its contained execution script. As the right flow chart shows, instruction block A and B are called as loop body. If the loop condition is false, re-execute the loop, otherwise the loop ends. When you need to repeat the same script, we often use the loop structure in programming.	



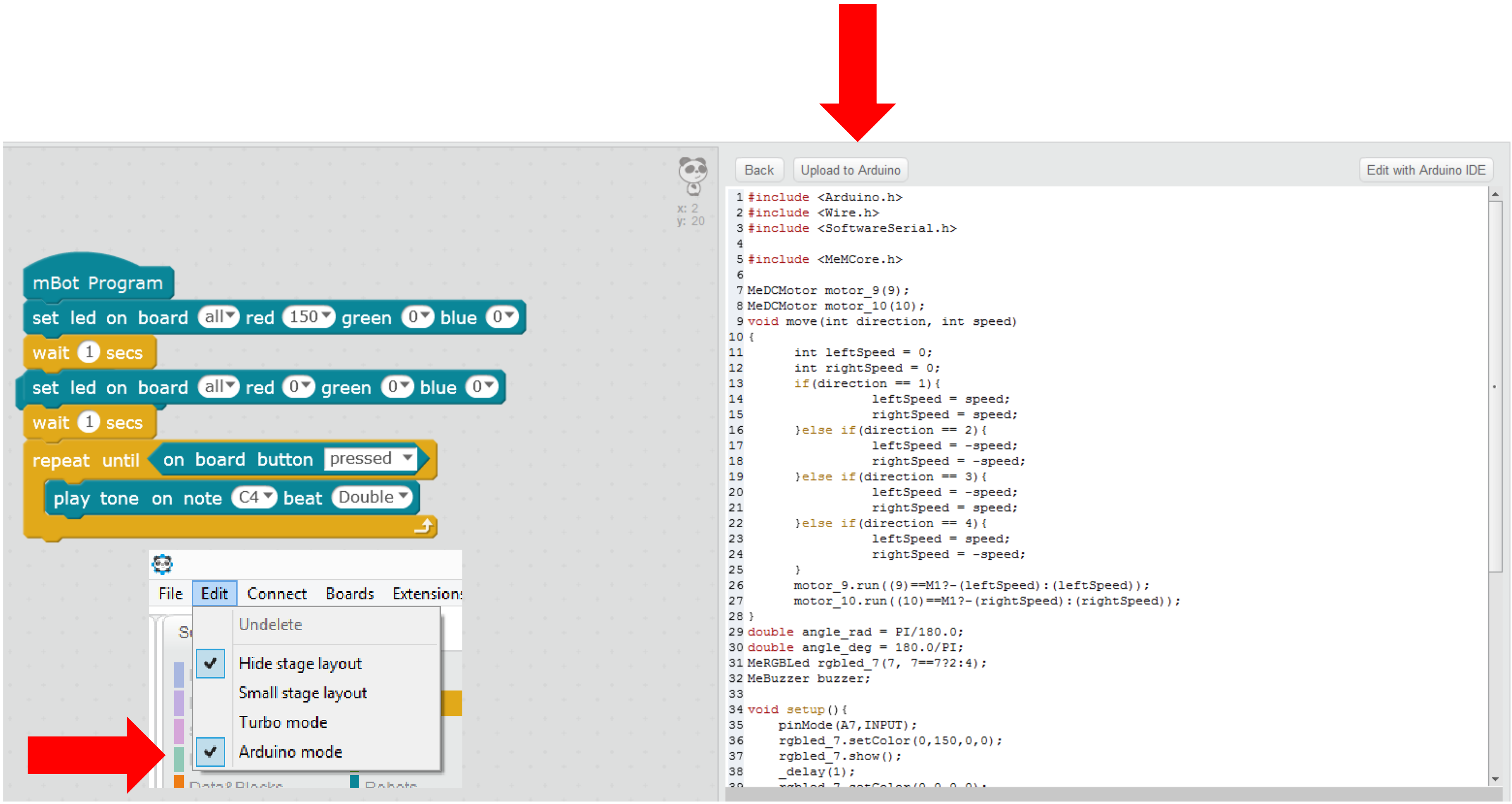
```
repeat until on board button pressed
    play tone on note C4 beat Double
```

Play a Tone Example 1



```
mBot Program
set led on board all red 150 green 0 blue 0
wait 1 secs
set led on board all red 0 green 0 blue 0
wait 1 secs
repeat until on board button pressed
    play tone on note C4 beat Double
```

Loading Programs to the Robot



```
mBot Program
set led on board all red 150 green 0 blue 0
wait 1 secs
set led on board all red 0 green 0 blue 0
wait 1 secs
repeat until on board button pressed
   play tone on note C4 beat Double
```

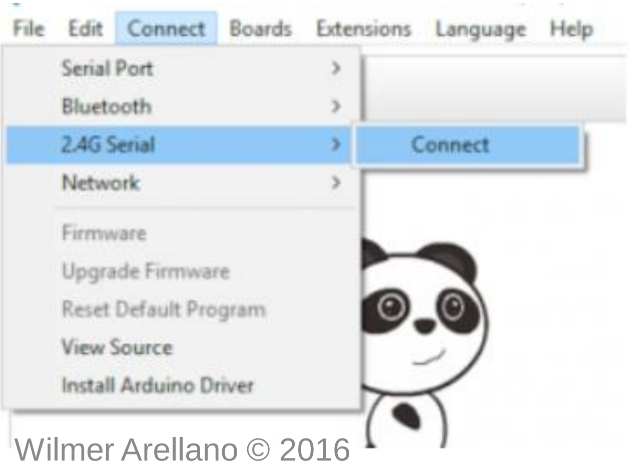
```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

#include <MeMCore.h>

MeDCMotor motor_9(9);
MeDCMotor motor_10(10);
void move(int direction, int speed)
{
      int leftSpeed = 0;
      int rightSpeed = 0;
      if(direction == 1){
                leftSpeed = speed;
                rightSpeed = speed;
      }else if(direction == 2){
                leftSpeed = -speed;
                rightSpeed = -speed;
      }else if(direction == 3){
                leftSpeed = -speed;
                rightSpeed = speed;
      }else if(direction == 4){
                leftSpeed = speed;
                rightSpeed = -speed;
      }
      motor_9.run((9)==M1?-(leftSpeed):(leftSpeed));
      motor_10.run((10)==M1?-(rightSpeed):(rightSpeed));
}
double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
MeRGBLed rgbled_7(7, 7==7?2:4);
MeBuzzer buzzer;

void setup(){
    pinMode(A7,INPUT);
    rgbled_7.setColor(0,150,0,0);
    rgbled_7.show();
    _delay(1);
    rgbled_7.setColor(0,0,0,0);
```

Connect Your Robot Through 2.4G Modules

- In this mode, the program runs in the computer which sends instructions to the robot on the actions to perform.
- This is an interesting feature for applications as the remote control operation illustrated bellow.
- For most applications it is not the preferred way of operating the robot.

Connect Your Robot Through 2.4G Modules

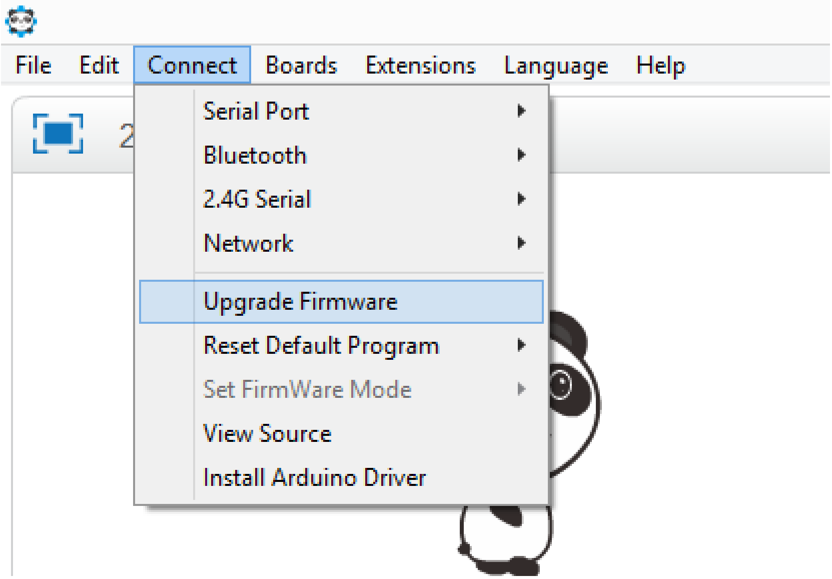
- The 2.4G module set comes with a electronic module and a USB dongle. If your are using the dongle first time, you need to connect the 2.4G electronic module to your robot; power up your robot; press the button on top of the 2.4G module, and then plug the dongle to your computer. When the LED light on the module turns from blinking to steady lighting, it means the 2.4G module is paired with the dongle.
- On the software side, after pairing the dongle you need to select Connect->2.4G Serial->Connect in the software menu. When "Connected" is shown in the title bar, you know your mBot is connected to the computer.



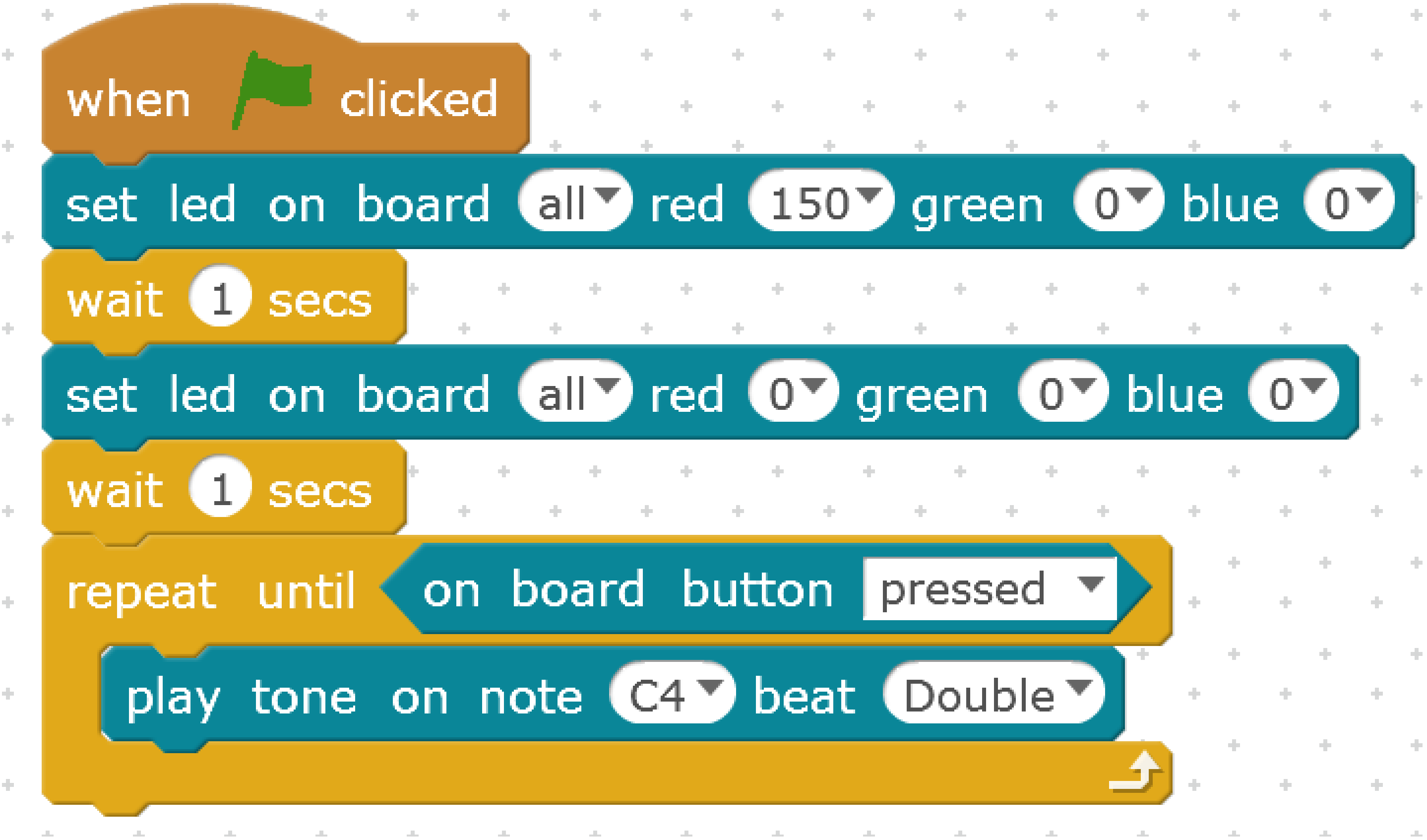
Wilmer Arellano © 2016

Connect Your Robot Through 2.4G Modules

- Connect with USB cable
- Upgrade Firmware
- Close Program
- Turn off m-Robot
- Open Program
- Turn On Robot
- Connect with 2.4G



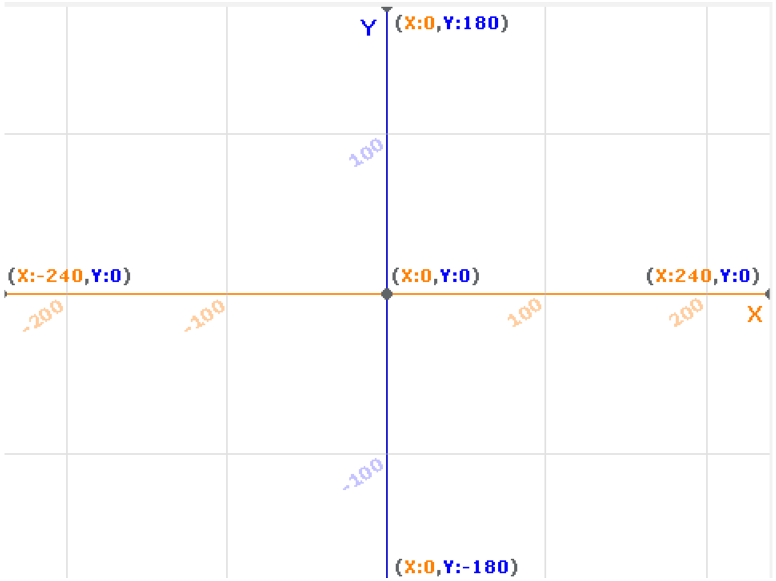
Play a Tone Example 2



```
when ⚑ clicked
set led on board all red 150 green 0 blue 0
wait 1 secs
set led on board all red 0 green 0 blue 0
wait 1 secs
repeat until <on board button pressed>
    play tone on note C4 beat Double
```

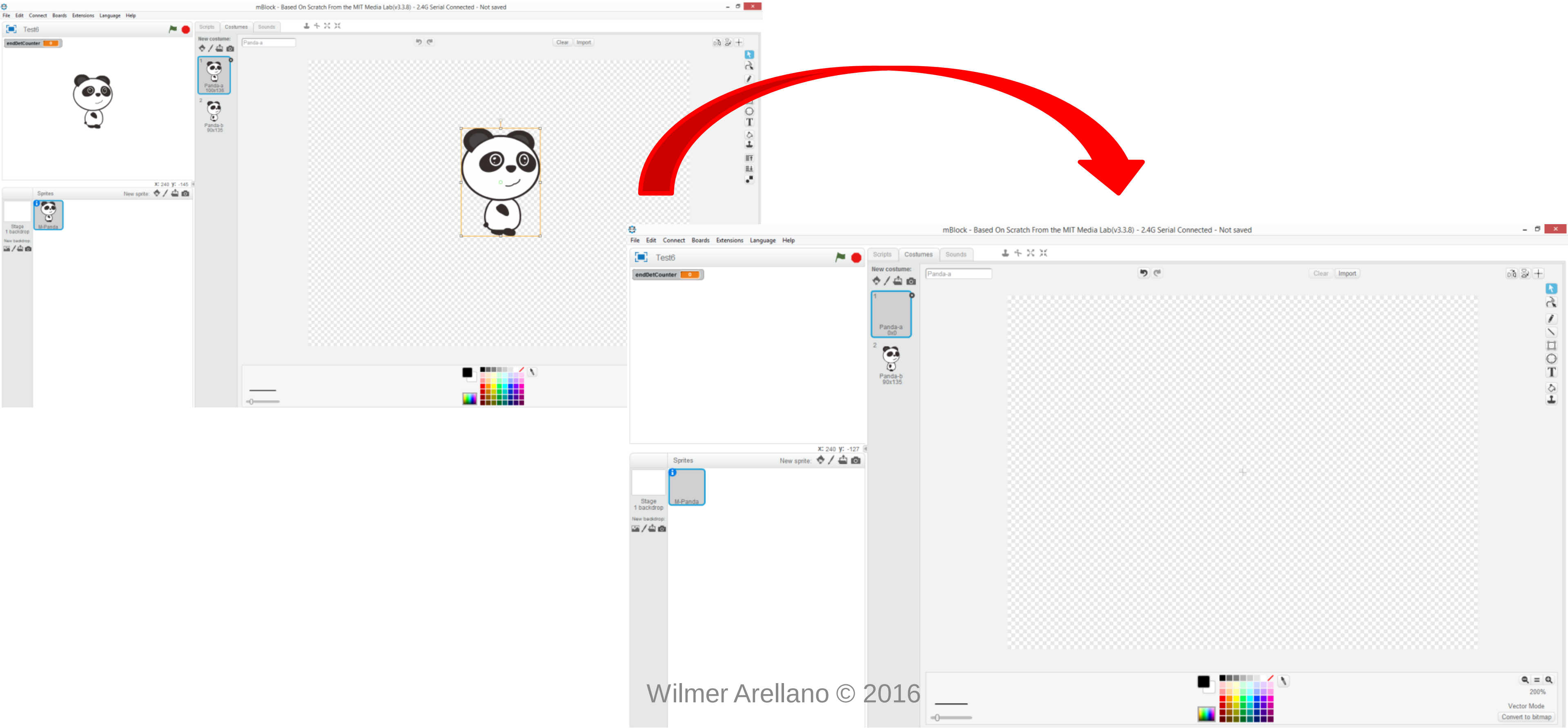
Remote Control

- Select the xy-grid backdrop

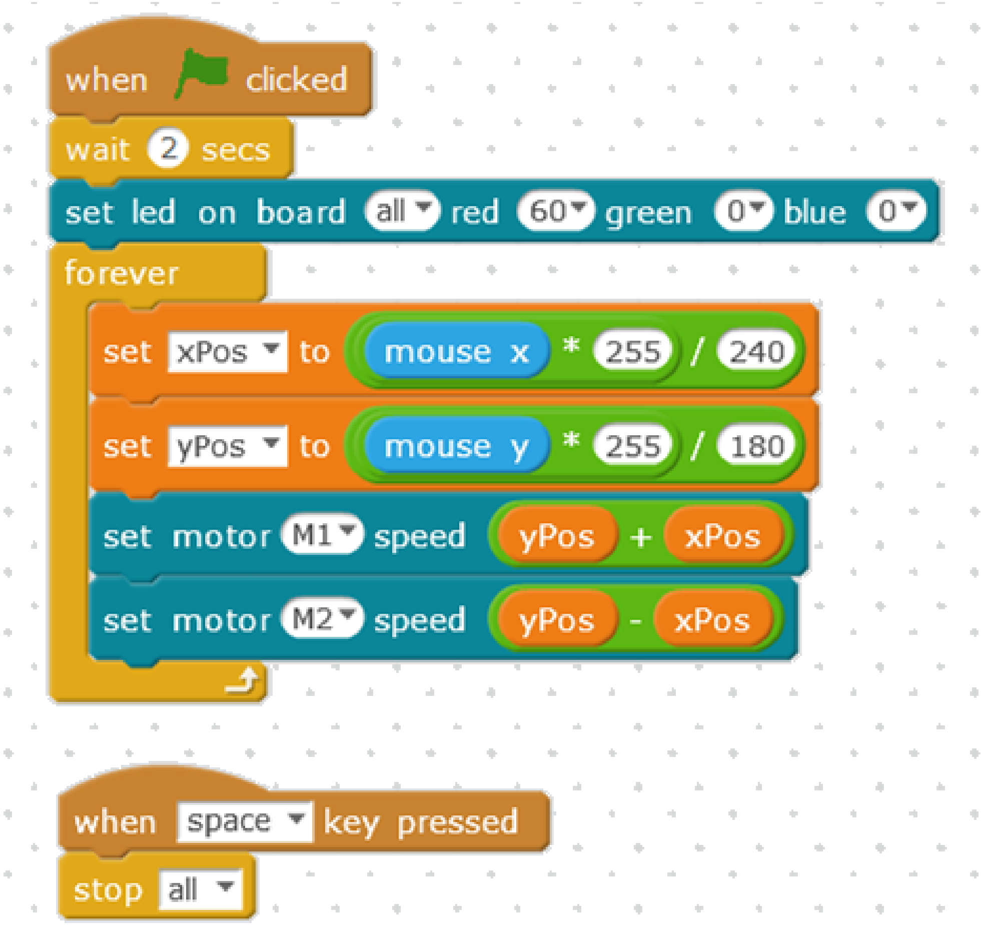


Remote Control

- Delete the image on costume 1 of the Panda



Remote Control



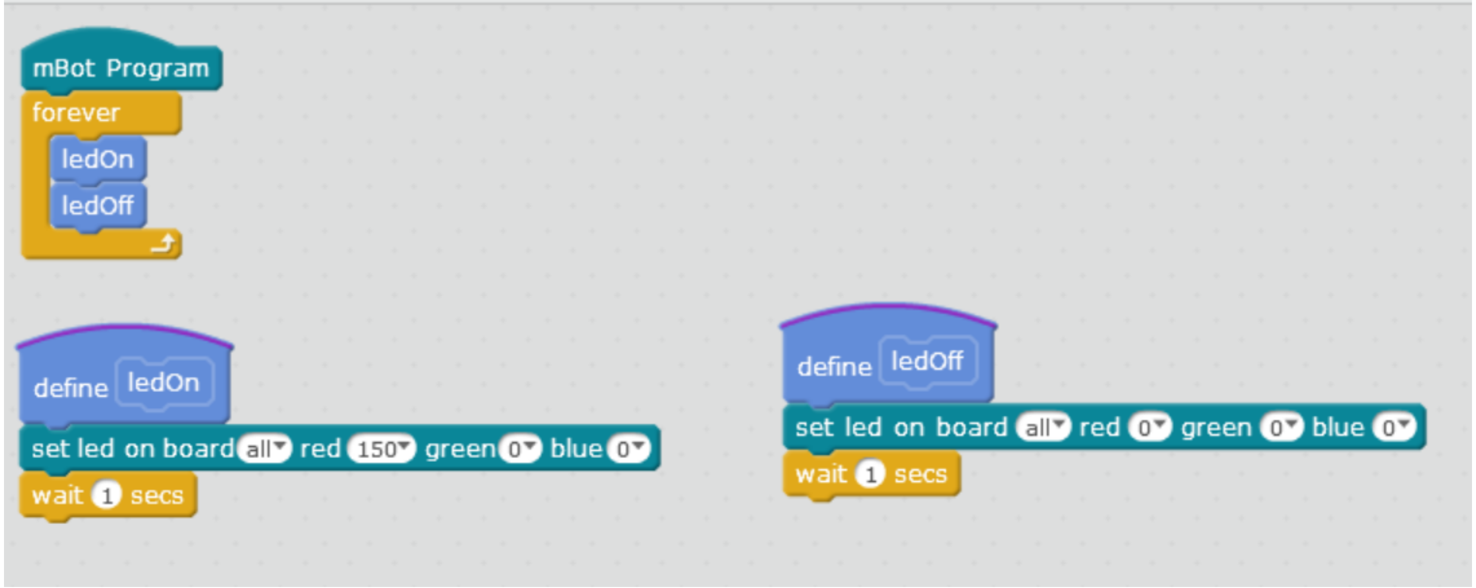
```
when ⚑ clicked
wait (2) secs
set led on board (all ▾) red (60 ▾) green (0 ▾) blue (0 ▾)
forever
    set [xPos ▾] to (((mouse x) * (255)) / (240))
    set [yPos ▾] to (((mouse y) * (255)) / (180))
    set motor (M1 ▾) speed ((yPos) + (xPos))
    set motor (M2 ▾) speed ((yPos) - (xPos))

when [space ▾] key pressed
stop [all ▾]
```

Functions

- A function is a group of statements that together perform a task.
- Functions are a utilized to make programs:
  - Easier to understand
  - Easier to troubleshoot
- As it can be observed in next slide, the main program can be written as macro instructions.
- These macro instructions will be executed one by one, as described in the definitions.
- It may be convenient to program a robot using functions, one per robot task.

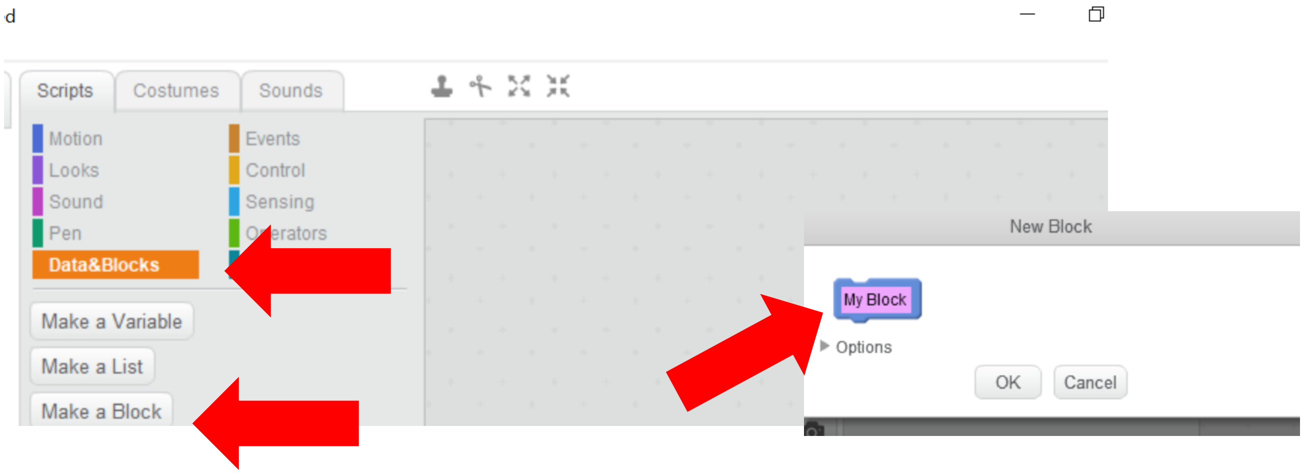
Functions

- This example is for the mBlock and the onboard LEDs



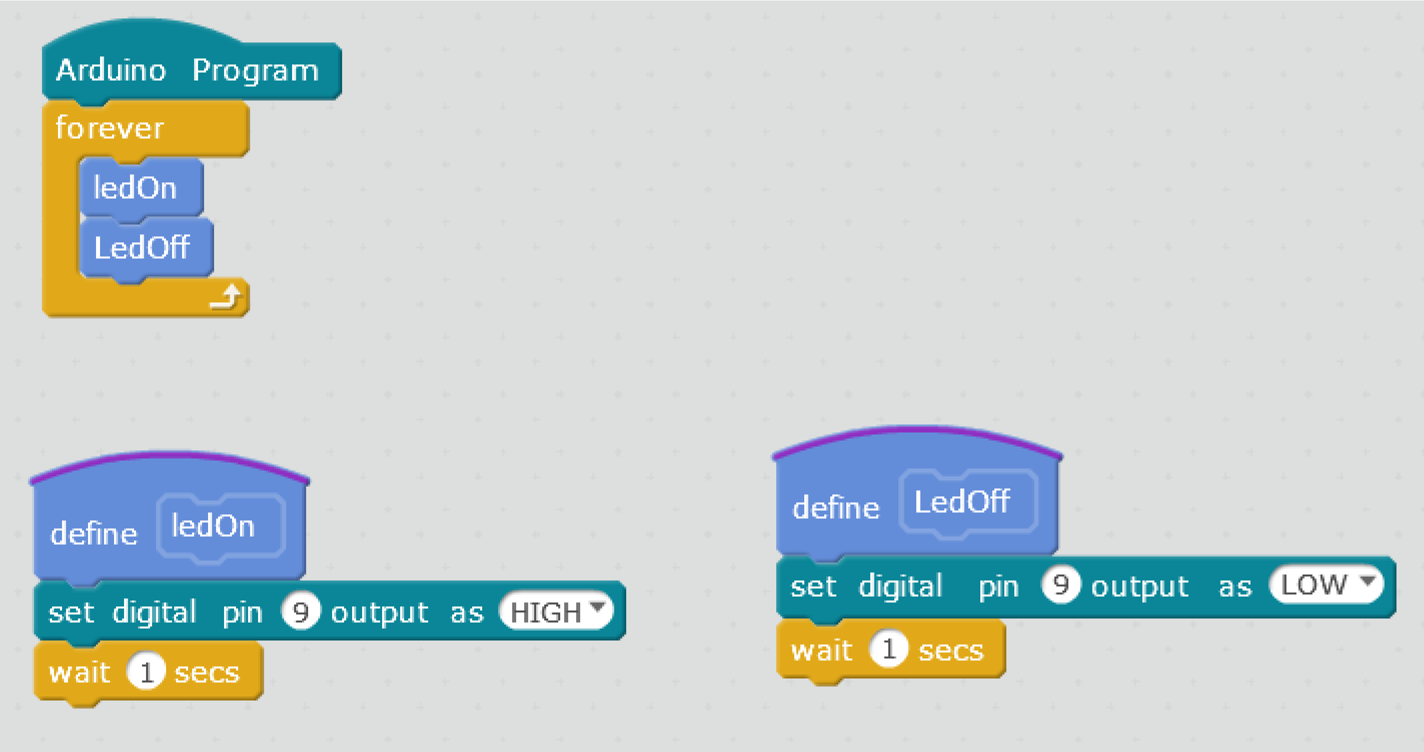
```
mBot Program
forever
    ledOn
    ledOff

define ledOn
set led on board all red 150 green 0 blue 0
wait 1 secs

define ledOff
set led on board all red 0 green 0 blue 0
wait 1 secs
```

Functions


d
Scripts
Costumes
Sounds
Motion
Looks
Sound
Pen
Data&Blocks
Events
Control
Sensing
Operators
Make a Variable
Make a List
Make a Block
New Block
My Block
Options
OK
Cancel

Functions

- This example is for an Arduino board with an LED connected to pin 9.

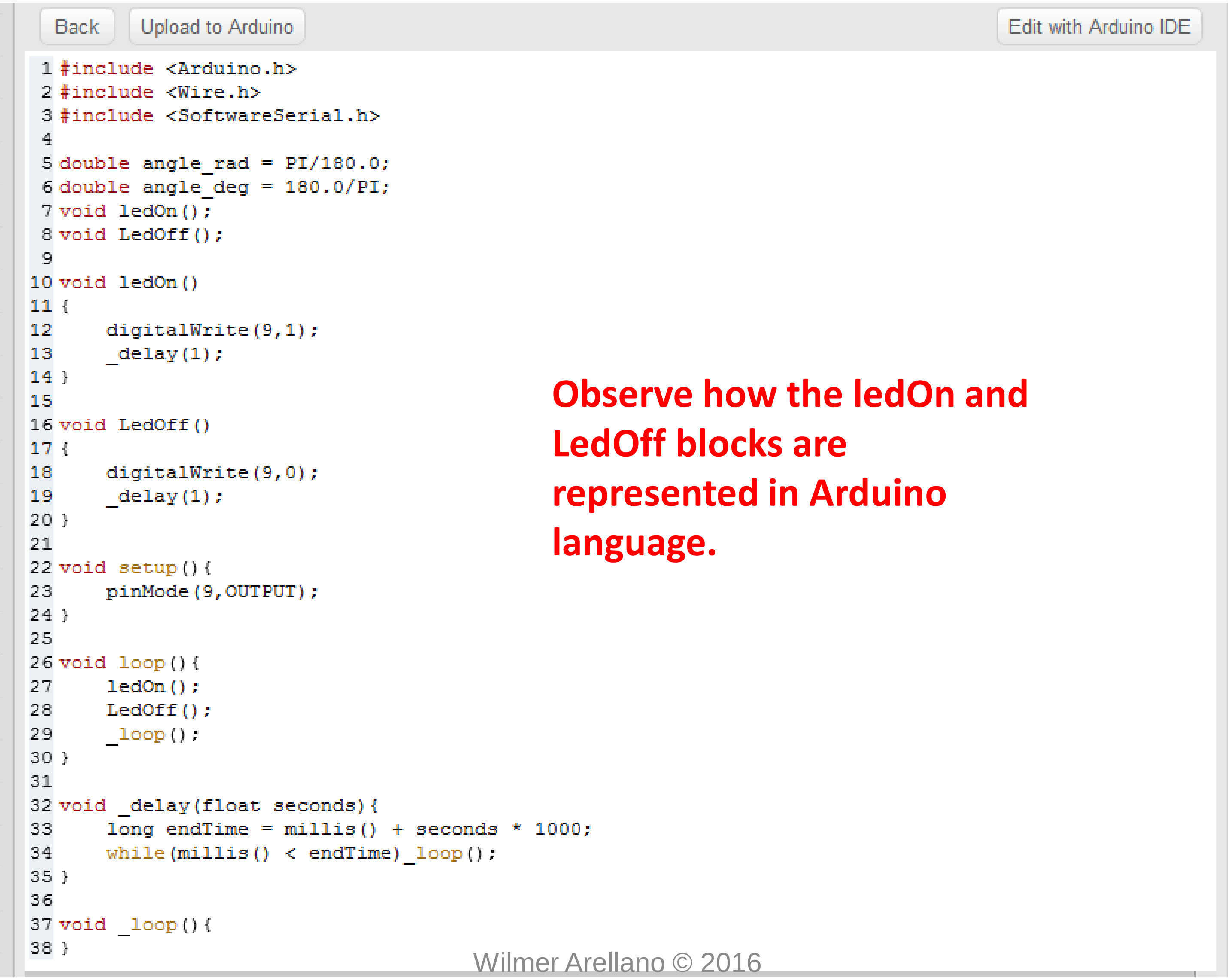


```
Arduino Program
forever
    ledOn
    LedOff
```

```
define ledOn
set digital pin 9 output as HIGH
wait 1 secs
```

```
define LedOff
set digital pin 9 output as LOW
wait 1 secs
```

Functions



```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
void ledOn();
void LedOff();

void ledOn()
{
    digitalWrite(9,1);
    _delay(1);
}

void LedOff()
{
    digitalWrite(9,0);
    _delay(1);
}

void setup(){
    pinMode(9,OUTPUT);
}

void loop(){
    ledOn();
    LedOff();
    _loop();
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}

void _loop(){
}
```

Observe how the ledOn and LedOff blocks are represented in Arduino language.

Code typed directly on Arduino Ide

```
void ledOn()
{
    digitalWrite(9,1);
    _delay(1);
}

void LedOff()
{
    digitalWrite(9,0);
    _delay(1);
}

void setup(){
    pinMode(9,OUTPUT);
}

void loop(){
    ledOn();
    LedOff();
    _loop();
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}

void _loop(){
}
```

millis(): Returns the number of milliseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.

Other versions of the _loop() function

```
void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)
    _loop();

  void _delay(float seconds){
      long endTime = millis() + seconds * 1000;
      while(millis() < endTime){
      _loop();
      }
```

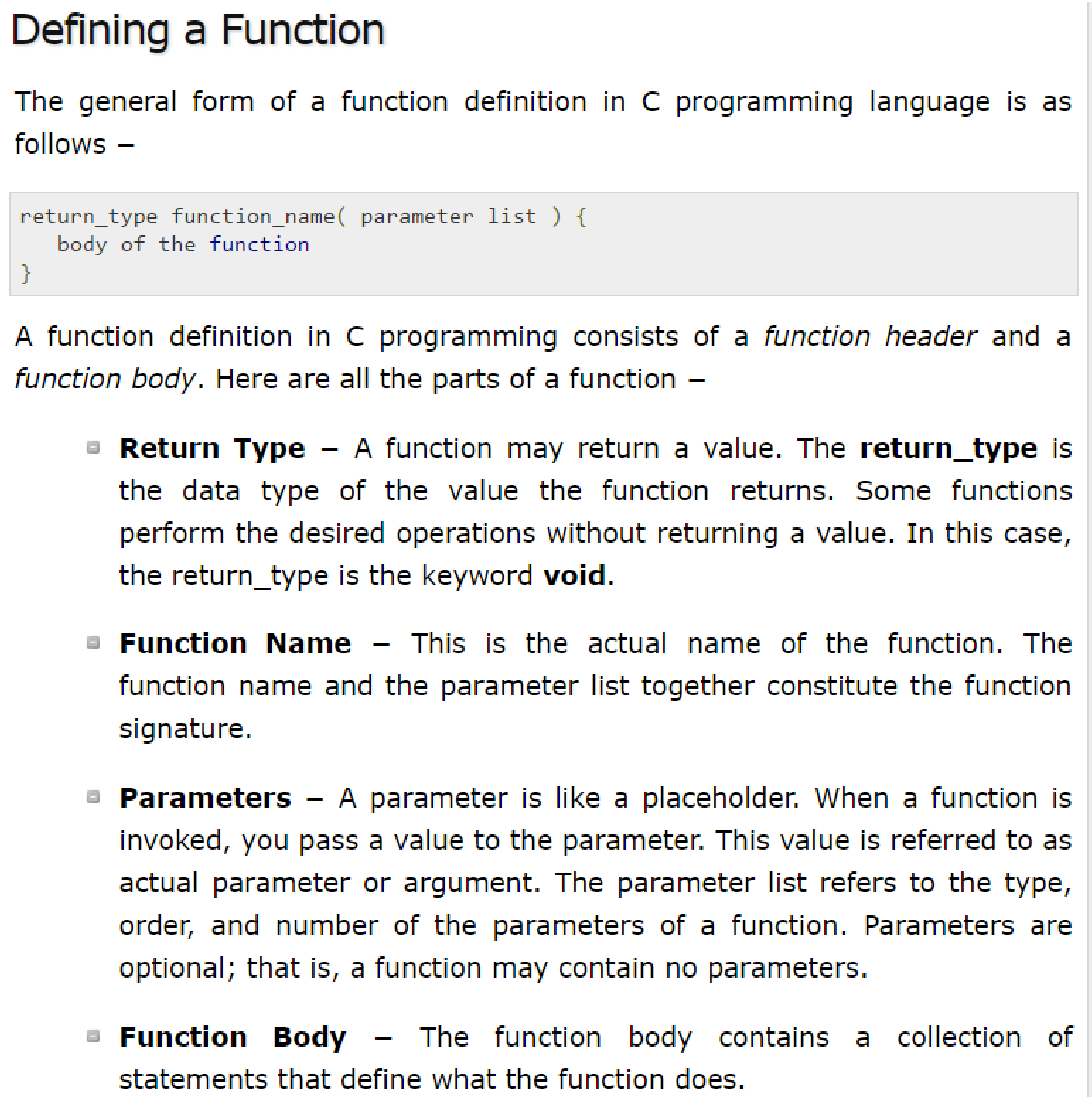
Simplified code on Arduino Ide

```
void ledOn()
{
   digitalWrite(9,1);
   delay(1000);
}

void LedOff()
{
   digitalWrite(9,0);
   delay(1000);
}

void setup(){
   pinMode(9,OUTPUT);
}

void loop(){
   ledOn();
   LedOff();
}
```

Advantages of the _loop() function

```
int frequency;
void ledOn()
{
    digitalWrite(9,1);
    frequency = 1000;
    _delay(1);
}

void LedOff()
{
    digitalWrite(9,0);
    frequency = 500;
    _delay(1);
}

void setup(){
    pinMode(9,OUTPUT);
}

void loop(){
    ledOn();
    LedOff();
    _loop();
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}

void _loop(){
  tone(8, frequency);
  delay(100);
  noTone(8);
  delay(100);
}
```

You can execute instructions while waiting



Defining a Function

The general form of a function definition in C programming language is as follows −

```
return_type function_name( parameter list ) {
   body of the function
}
```

A function definition in C programming consists of a *function header* and a *function body*. Here are all the parts of a function −

- **Return Type** − A function may return a value. The **return_type** is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword **void**.

- **Function Name** − This is the actual name of the function. The function name and the parameter list together constitute the function signature.

- **Parameters** − A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.

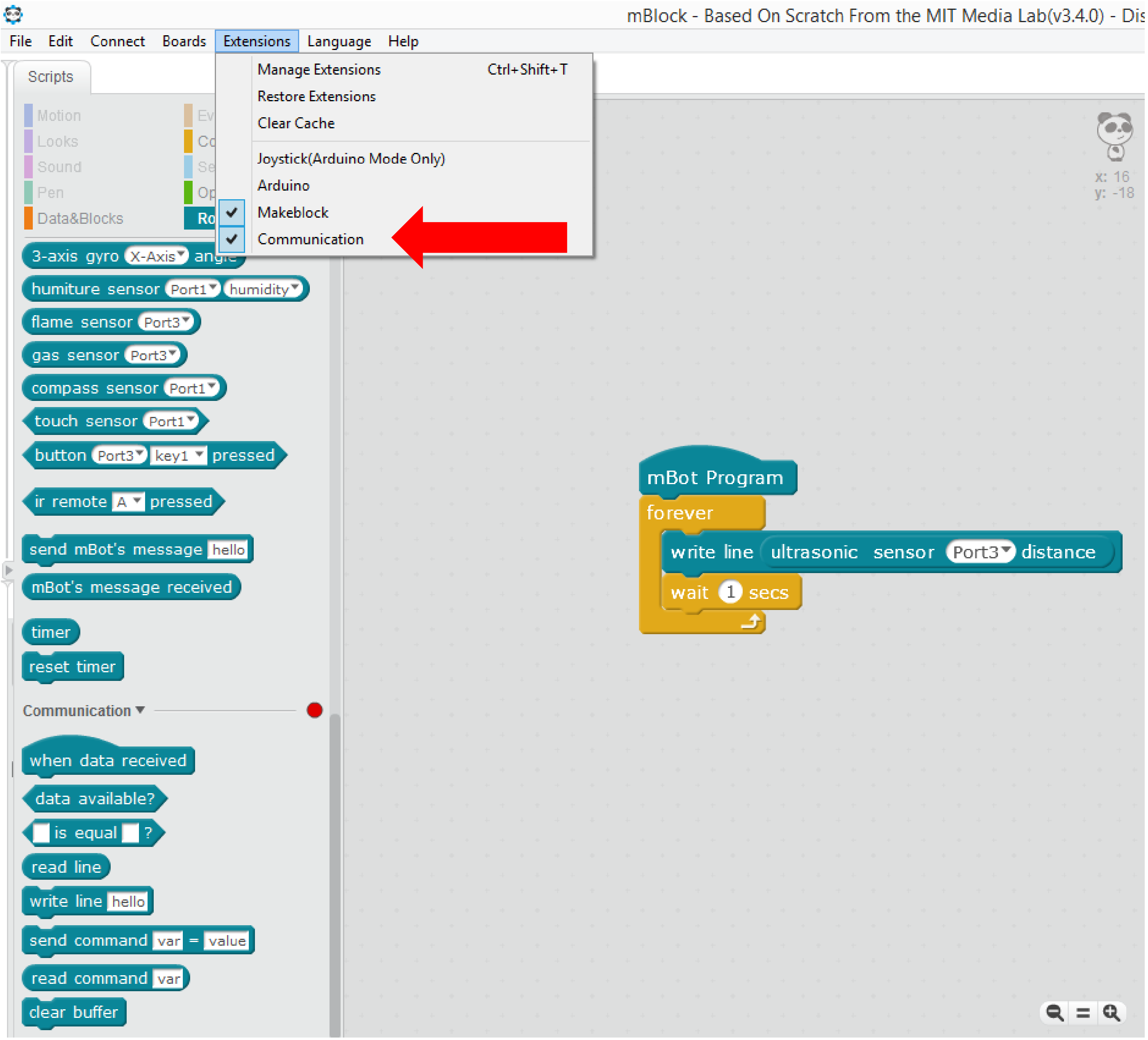
- **Function Body** − The function body contains a collection of statements that define what the function does.

```
/*
 * “Hello World!”
 * This is the Hello World! for Arduino.
 * It shows how to send data to the computer
 */
void setup()                      // run once, when the sketch starts
{
  Serial.begin(9600);             // set up Serial library at 9600 bps

  Serial.println("Is anybody out there?");  // prints phrase with ending
   line break
}
void loop()                         // run over and over again
{
                                    // do nothing!
}
// After sending program to the Arduino, press Reset button on the board
   and watch Serial monitor
```

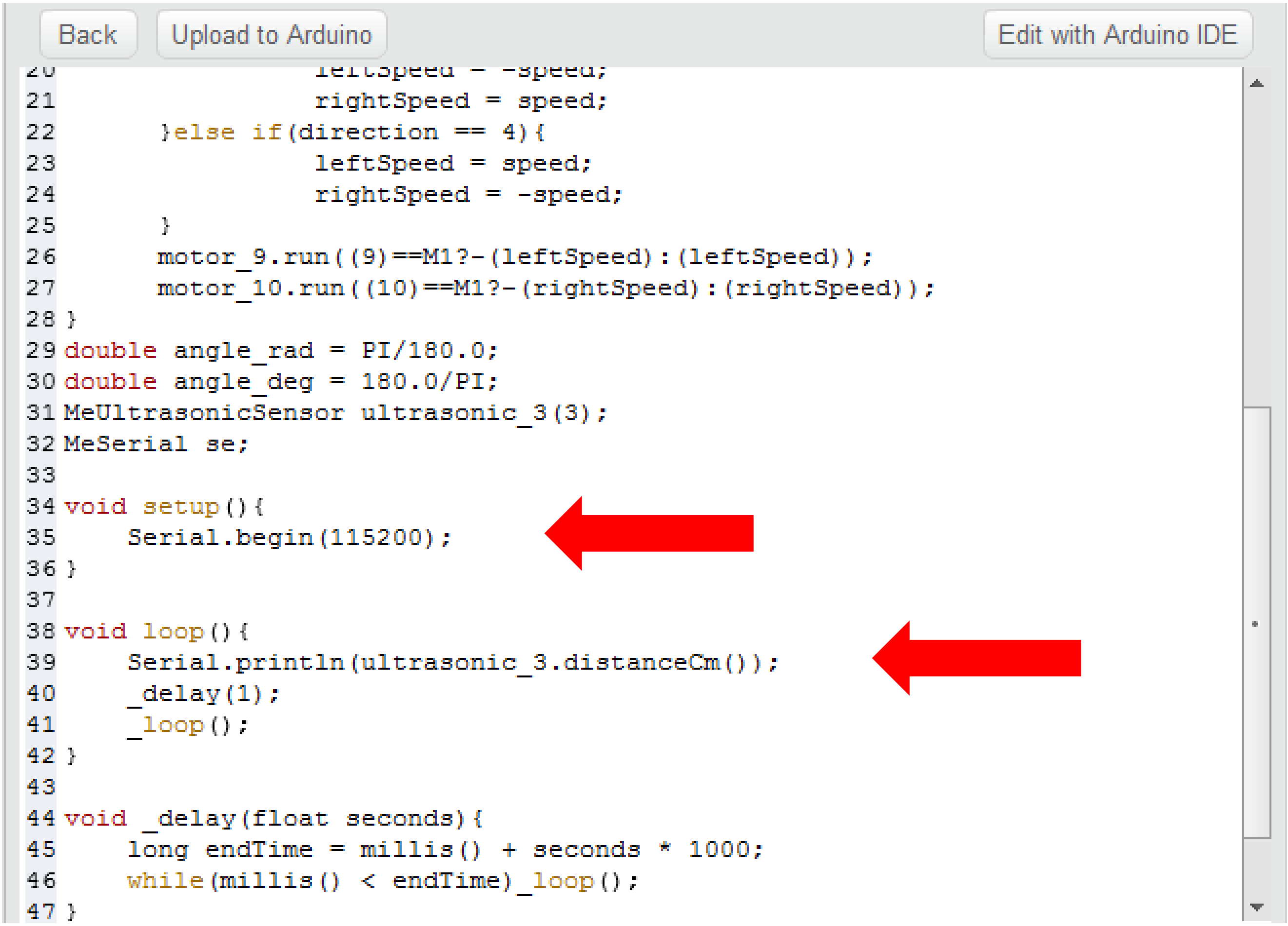
```
/*
Function Example
 */
 float quotient;
 void setup(){
   Serial.begin(9600);
   quotient = divide1(7, 2);
   Serial.print("divide1 ");
   Serial.println (quotient);
   Serial.print("divide2 ");
   Serial.println (divide2(7, 2));
 }
 void loop(){
 }

 /* This function will return 3.00 even though
 it has a float type. This is because calculation is
based on integers (integer math) */
 float divide1(int dividend, int divisor){
   float result;
   result = dividend/divisor;
   return result;
 }

 /* This function will return 3.50 the cast (type)with
argument float forces float math */
 float divide2(int dividend, int divisor){
   float result;
   result = (float)dividend/divisor;
   return result;
 }
```

Testing The Ultrasonic sensor

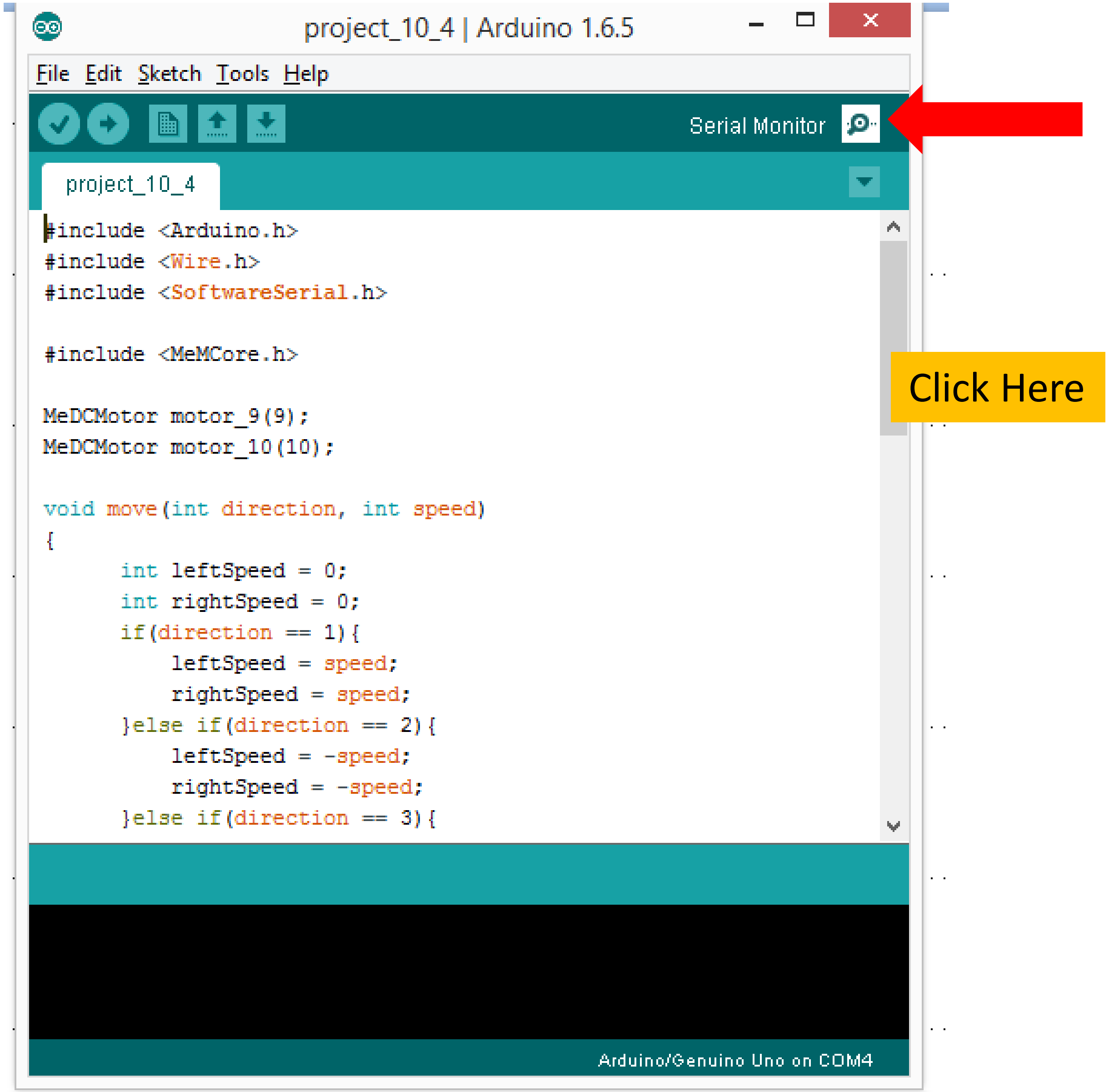
- This program will allow to send distance measured by the ultrasonic sensor to a computer.
- A USB cable must be used to connect the robot to de computer.
- Port must be selected in mBlock and in Arduino IDE.



```
mBot Program
forever
    write line ultrasonic sensor Port3 distance
    wait 1 secs
```



```
                leftSpeed = -speed;
                rightSpeed = speed;
      }else if(direction == 4){
                leftSpeed = speed;
                rightSpeed = -speed;
      }
      motor_9.run((9)==M1?-(leftSpeed):(leftSpeed));
      motor_10.run((10)==M1?-(rightSpeed):(rightSpeed));
}
double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
MeUltrasonicSensor ultrasonic_3(3);
MeSerial se;

void setup(){
    Serial.begin(115200);
}

void loop(){
    Serial.println(ultrasonic_3.distanceCm());
    _delay(1);
    _loop();
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}
```

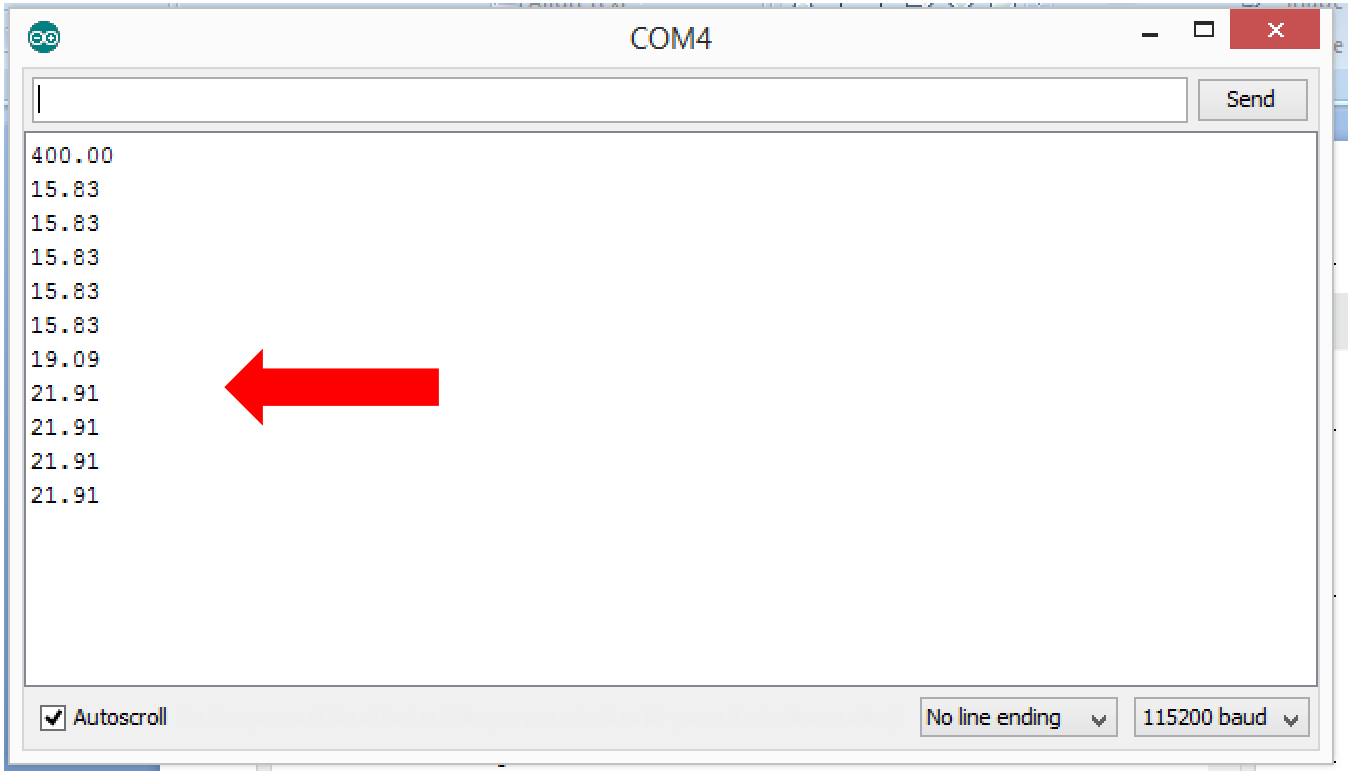


project_10_4 | Arduino 1.6.5
File Edit Sketch Tools Help
Serial Monitor
project_10_4
Arduino/Genuino Uno on COM4

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

#include <MeMCore.h>

MeDCMotor motor_9(9);
MeDCMotor motor_10(10);

void move(int direction, int speed)
{
      int leftSpeed = 0;
      int rightSpeed = 0;
      if(direction == 1){
          leftSpeed = speed;
          rightSpeed = speed;
      }else if(direction == 2){
          leftSpeed = -speed;
          rightSpeed = -speed;
      }else if(direction == 3){
```

Click Here

Distance readings will be sent to computer



```
400.00
15.83
15.83
15.83
15.83
15.83
19.09
21.91
21.91
21.91
21.91
```

☑ Autoscroll

No line ending

115200 baud

Select right speed

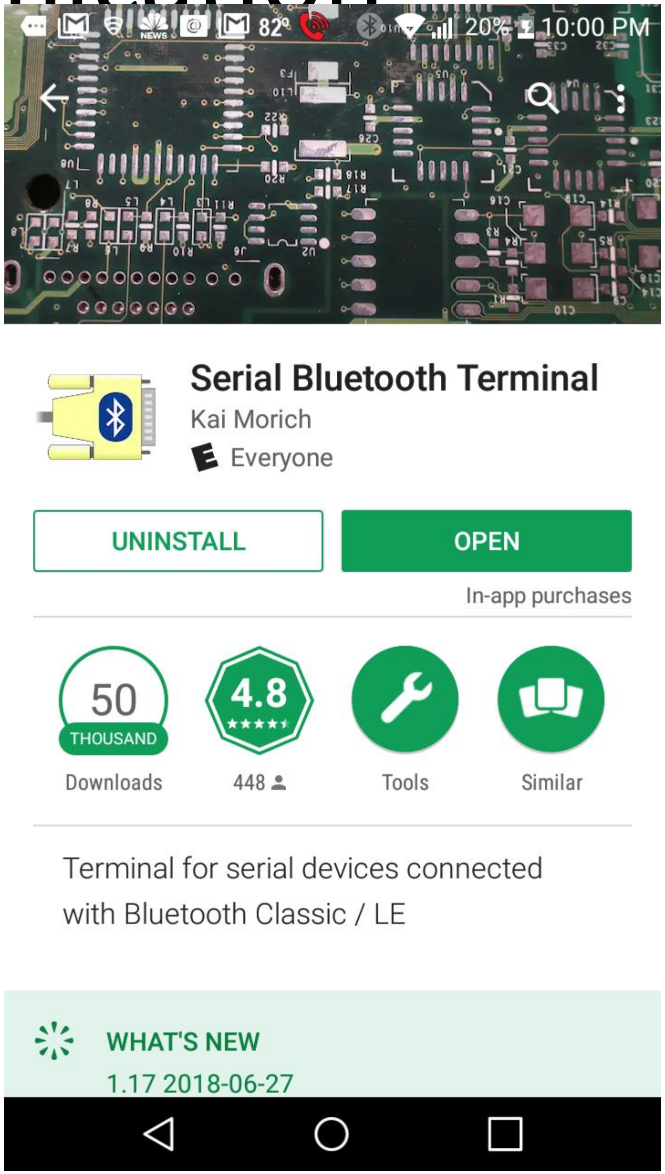
Wilmer Arellano © 2016

Bluetooth Communication

- Bluetooth communication is easy with the mBot
- We will illustrate with a program that sends and receives text.
- Connect the Bluetooth module to the mBot

Bluetooth Communication

- We will explain Bluetooth communication for the Android App
  - Serial Bluetooth Terminal
- Install the App in your phone



Serial Bluetooth Terminal

Kai Morich

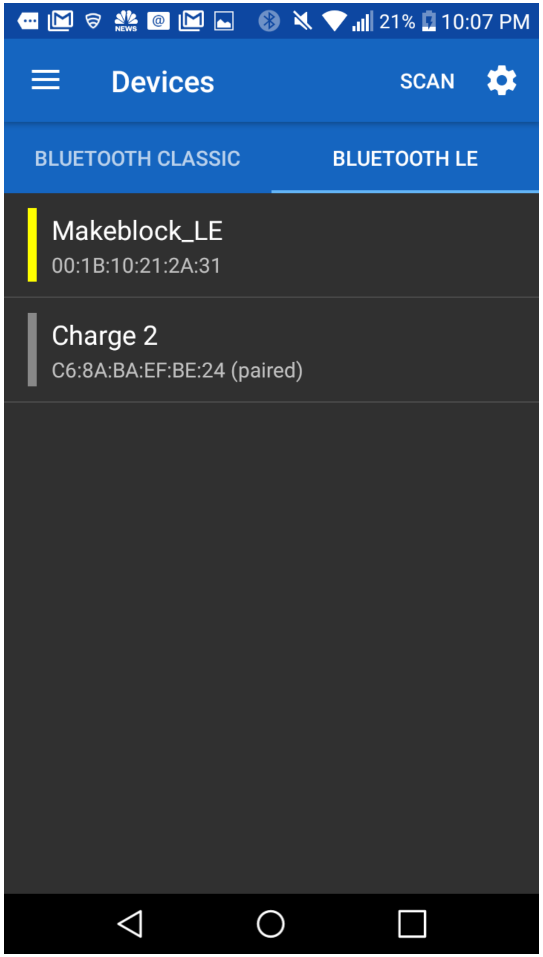
Terminal for serial devices connected with Bluetooth Classic / LE

WHAT'S NEW

1.17 2018-06-27

Bluetooth Communication

- Navigate to Devices
- Select Bluetooth LE
- Click Scan
- Press and hold Mackeblock for further configuration

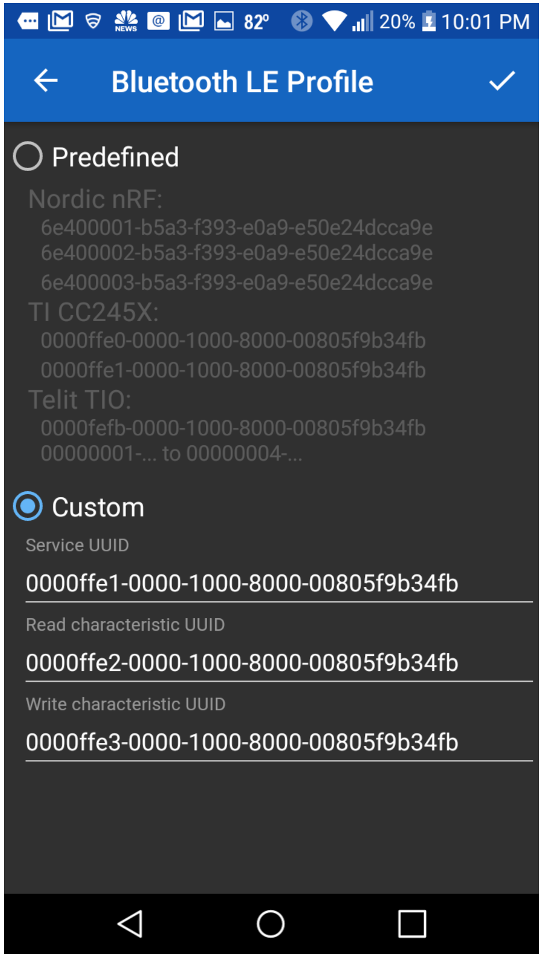


Makeblock_LE
00:1B:10:21:2A:31

Charge 2
C6:8A:BA:EF:BE:24 (paired)

Bluetooth Communication

- Select Custom
- Press and hold each of the three bottom fields and select to the image



○ Predefined

Nordic nRF:
6e400001-b5a3-f393-e0a9-e50e24dcca9e
6e400002-b5a3-f393-e0a9-e50e24dcca9e
6e400003-b5a3-f393-e0a9-e50e24dcca9e

TI CC245X:
0000ffe0-0000-1000-8000-00805f9b34fb
0000ffe1-0000-1000-8000-00805f9b34fb

Telit TIO:
0000fefb-0000-1000-8000-00805f9b34fb
00000001-... to 00000004-...

● Custom

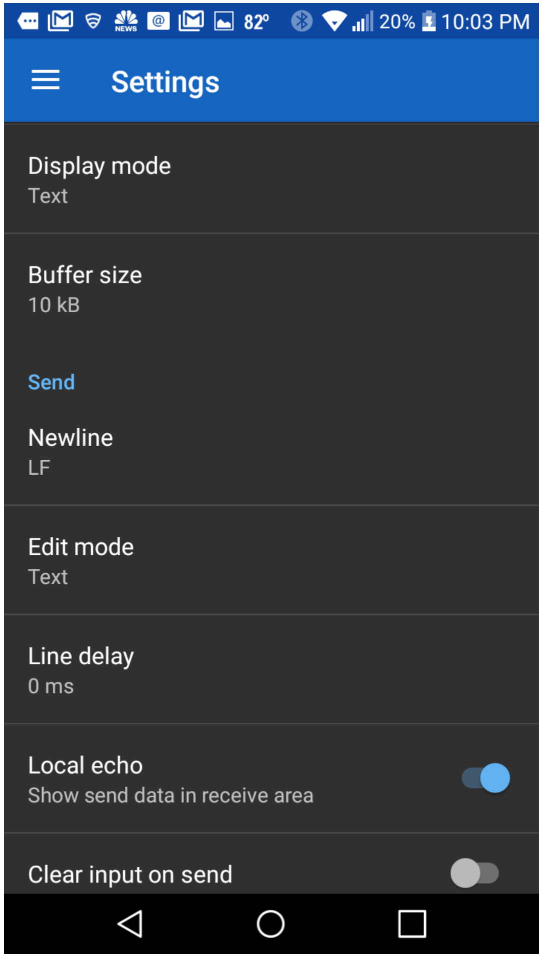
Service UUID
0000ffe1-0000-1000-8000-00805f9b34fb

Read characteristic UUID
0000ffe2-0000-1000-8000-00805f9b34fb

Write characteristic UUID
0000ffe3-0000-1000-8000-00805f9b34fb

Bluetooth Communication

- Navigate to Settings
- Press and hold Send and select “LF”



82° 20% 10:03 PM

Settings

Display mode
Text

Buffer size
10 kB

Send

Newline
LF

Edit mode
Text

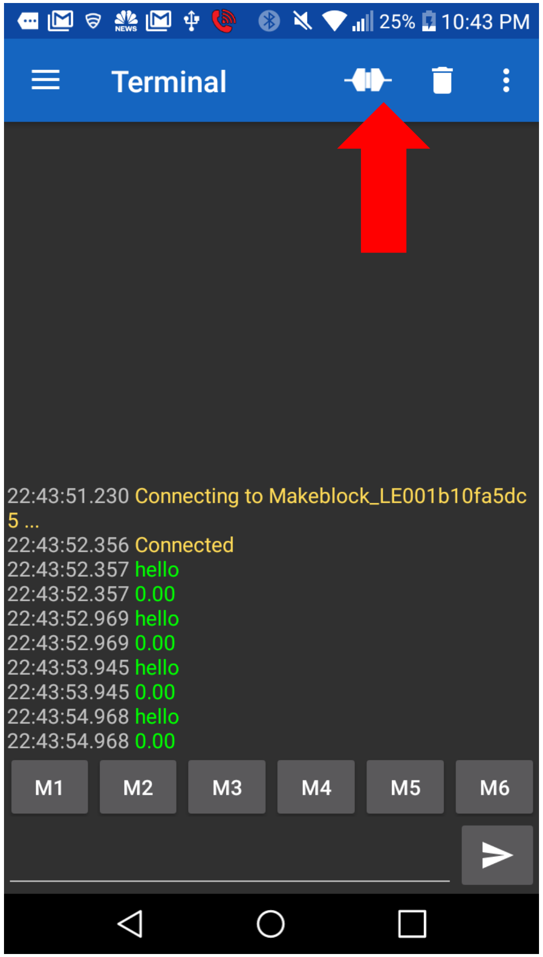
Line delay
0 ms

Local echo
Show send data in receive area

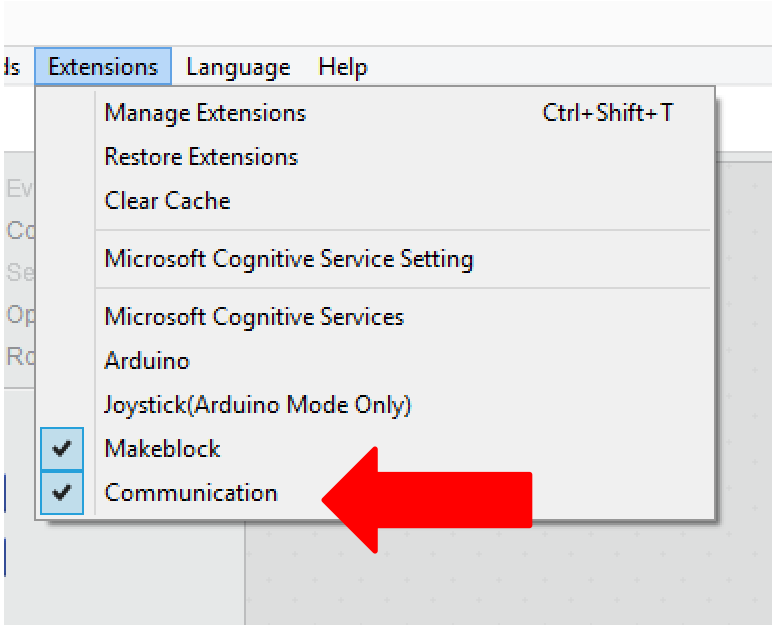
Clear input on send

Bluetooth Communication

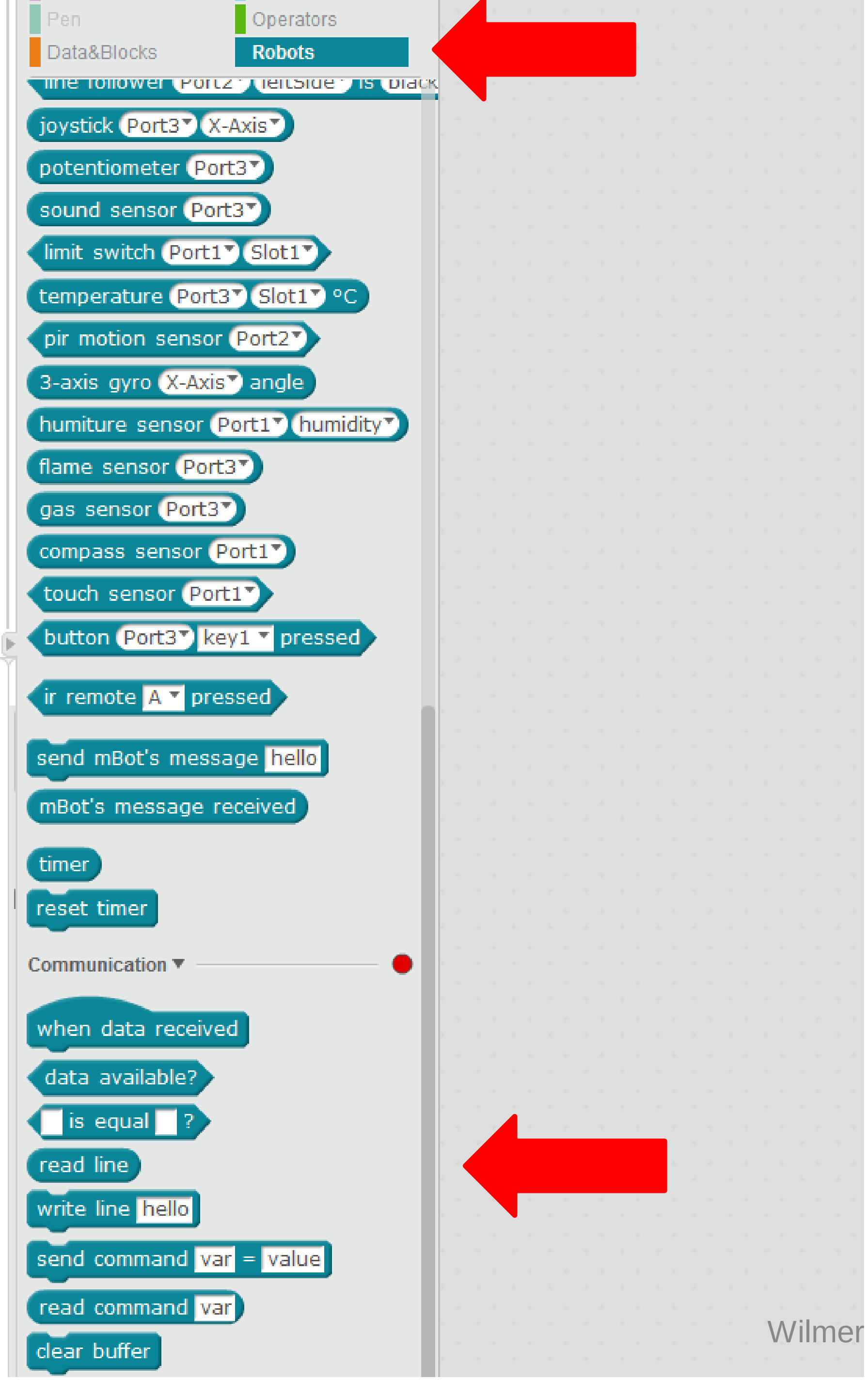
- Navigate to Terminal
- Click on Connect
- You are ready to start sending your commands



```
22:43:51.230 Connecting to Makeblock_LE001b10fa5dc
5 ...
22:43:52.356 Connected
22:43:52.357 hello
22:43:52.357 0.00
22:43:52.969 hello
22:43:52.969 0.00
22:43:53.945 hello
22:43:53.945 0.00
22:43:54.968 hello
22:43:54.968 0.00
```

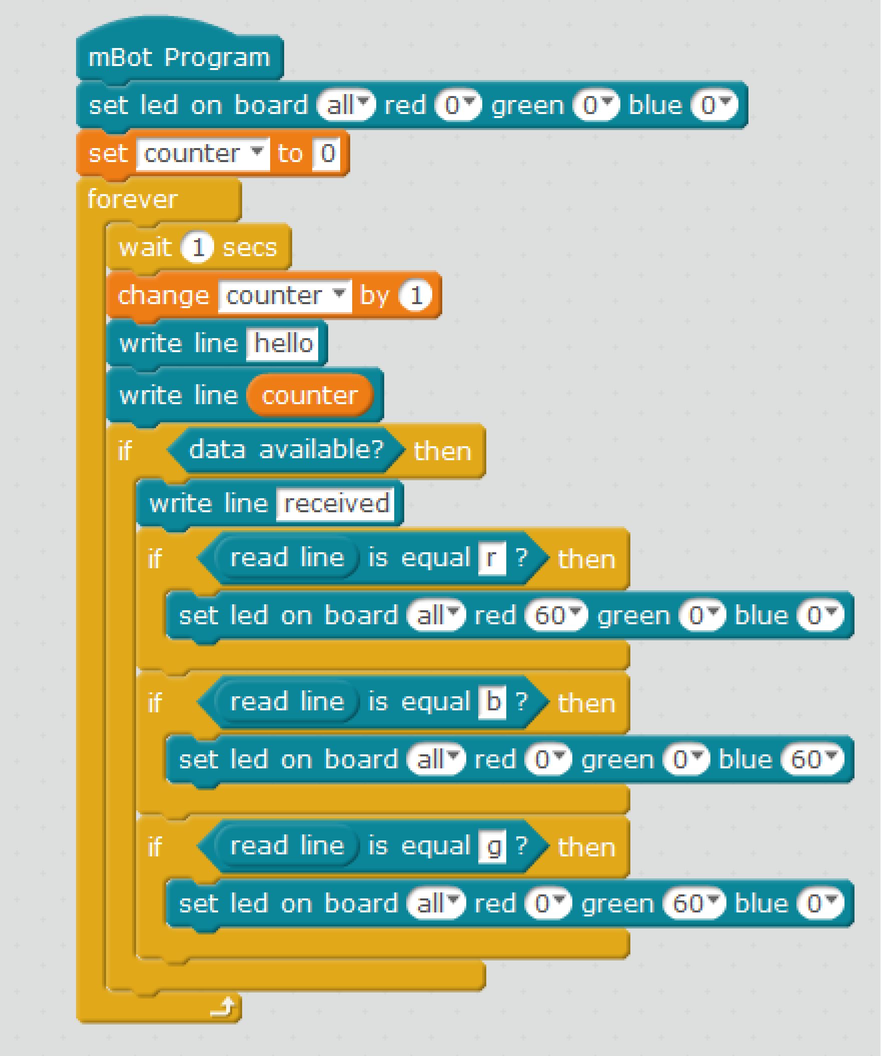


- Select The communication extension



- Click On the Robots section
- The same instructions used for serial communication with the computer are used to communicate with Bluetooth

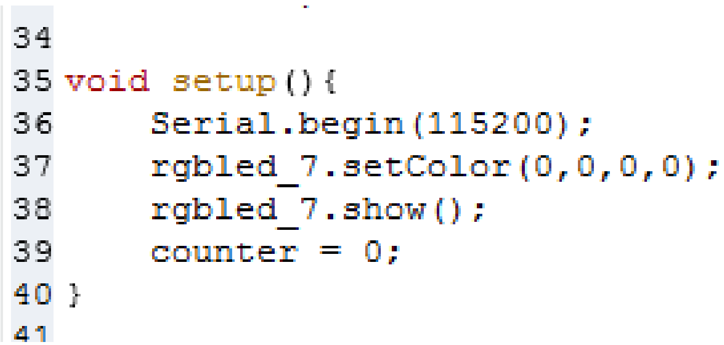
Bluetooth Communication



```
mBot Program
set led on board all red 0 green 0 blue 0
set counter to 0
forever
    wait 1 secs
    change counter by 1
    write line hello
    write line counter
    if data available? then
        write line received
        if read line is equal r ? then
            set led on board all red 60 green 0 blue 0
        if read line is equal b ? then
            set led on board all red 0 green 0 blue 60
        if read line is equal g ? then
            set led on board all red 0 green 60 blue 0
```

- Load the program on the left into the mBot
- This program will write "hello" and the value of counter in your cell phone
- The onboard LEDs will change color to red, green, or blue, depending on if you send "r", "g", or "b" from your telephone

Bluetooth Communication



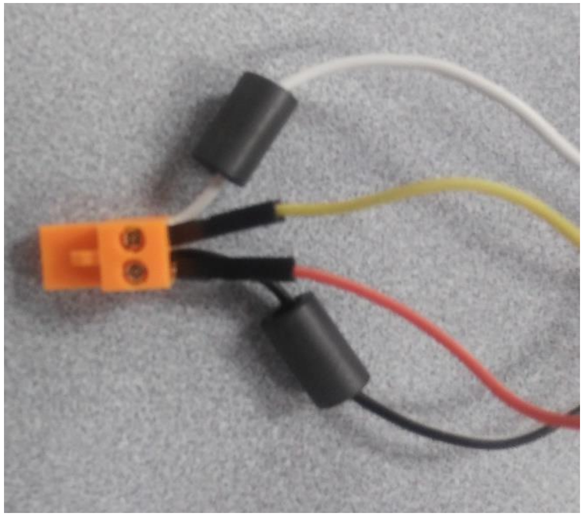
```
void setup(){
    Serial.begin(115200);
    rgbled_7.setColor(0,0,0,0);
    rgbled_7.show();
    counter = 0;
}
```

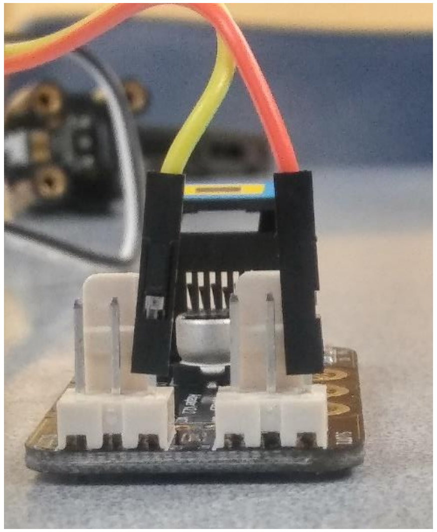
- On the Arduino side make sure that Serial.begin() has the argument 115200.

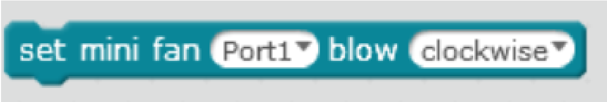
Gripper Connections

Gripper Connections

- Use to male-female jumper cables to connect the gripper to the RJ25 adapter
- Connect the RJ25 to one port. Use the mini fan block to control







```
set mini fan Port1▾ blow clockwise▾
```