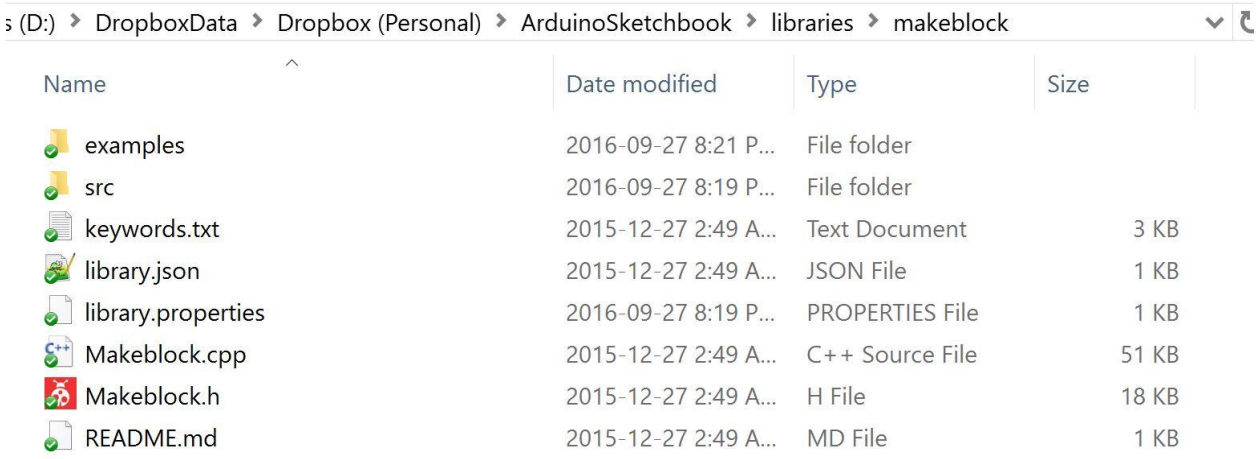
# Example Codes

mBot's main board is mCore. (mCore specifications)
Based on Arduino Uno. Compiler can select Uno.

**Install the Makeblock Library**
(Follow instructions on the GitHub page)
https://github.com/Makeblock-official/Makeblock-Libraries

The folder structure should look like this:




**Mac Driver**
http://blog.sengotta.net/signed-mac-os-driver-for-winchiphead-ch340-serial-bridge/

Pin Mapping (reference to Ardu-os-driver-for-winchiphead-ch340-serial-bridge/#comment-37545ino Uno)

```
A0 RJ25 plug 4 (default not-connected)
A1 RJ25 plug 4 (default not-connected)
A2 RJ25 plug 3 ultrasonic
A3 RJ25 plug 3 ultrasonic
A6 light sensor
A7 button
D2 IR RCV
D3 IR TX
D4 DIR2 - direction motor2
D5 PWM2 - pwm motor2
D6 PWM1 - pwn motor1
D7 DIR1 - direction motor1
D8 buzzer
D9 RJ25 plug 2 linefollower
D10 RJ25 plug 2 linefollower
D11 RJ25 plug 1 (default not-connected)
D12 RJ25 plug 1 (default not-connected)
D13 2 WS2812 RGB leds
```

PORT mapping of the Makeblock PORTs (telephone plugs). Start counting from 0.

```
MePort_Sig mePort[17] =
{
  { NC, NC }, { 11, 12 }, {  9, 10 }, { A2, A3 }, { A0, A1 },
  { NC, NC }, {  8, A6 }, { A7, 13 }, {  8, A6 }, {  6,  7 },
  {  5,  4 }, { NC, NC }, { NC, NC }, { NC, NC }, { NC, NC },
  { NC, NC },{ NC, NC },
};
```

## Output - RGB Led (On board)

http://wiki.makeblock.cc/library/docs/class_me_r_g_b_led.html

Sample code below will flash both LED in a looping three-step colour pattern

```
#include <MeMCore.h>

MeRGBLed led(0, 30);

void setup()
{
  led.setpin(13);
}

void loop()
{
  led.setColor(255, 255, 255); //Set both LED to White
  led.show();                  //Must use .show() to make new colour take effect.
  delay(500);

  led.setColorAt(0, 255, 0, 0); //Set LED0 (RGBLED1) (RightSide) to Red
  led.setColorAt(1, 0, 0, 255); //Set LED1 (RGBLED2) (LeftSide)  to Blue
  led.show();
  delay(500);

  led.setColorAt(0, 0, 0, 255); //Set LED0 (RGBLED1) (RightSide) to Blue
  led.setColorAt(1, 255, 0, 0); //Set LED1 (RGBLED2) (LeftSide)  to Red
  led.show();
  delay(500);
}
```

## Output - DC Motors (Left and right wheel)

http://wiki.makeblock.cc/library/docs/class_me_d_c_motor.html
http://learn.makeblock.com/en/Makeblock-library-for-Arduino/class_me_d_c_motor.html

Sample code below will run both motors in a looping pattern (Forward, Backward, Pause)
Recommended minimum speed is -50 or 50, value between -50 and 50 generate very low torque and will likely cause motor to stall.

```
#include <MeMCore.h>

MeDCMotor motor1(M1); //Motor1 is Left Motor
MeDCMotor motor2(M2); //Motor2 is Right Motor

void setup(){}

void loop()
{
  //motor.run() maximum speed is 255 to -255, 0 is stop
  motor1.run(-100); //Motor1 (Left)  forward is -negative
  motor2.run(100);  //Motor2 (Right) forward is +positive
  delay(500);

  motor1.run(100);  //Motor1 (Left)  backward is +positive
  motor2.run(-100); //Motor2 (Right) backward is -negative
  delay(500);

  motor1.stop(); //Stop Motor1
  motor2.stop(); //Stop Motor1
  delay(500);
}
```

## Output - Buzzer

http://learn.makeblock.com/en/Makeblock-library-for-Arduino/class_me_buzzer.html

Sample code below will sound the buzzer two times when the program is loaded.
The buzzer.tone() function blocks program execution for the duration of the sound

```
#include <MeMCore.h>

MeBuzzer buzzer;

void setup() {

  buzzer.tone(600, 1000);   //Buzzer sounds 600Hz for 1000ms
  delay(2000);              //Pause for 2000ms, Buzzer no sound
  buzzer.tone(1200, 1000);  //Buzzer sounds 1200Hz for 1000ms
  delay(2000);              //Pause for 2000ms, Buzzer no sound
}
```

```
void loop(){}
```

## Input - Black Line Finder Sensor

http://learn.makeblock.com/en/Makeblock-library-for-Arduino/class_me_line_follower.html#details

Sample code below will read the line finder sensor, write the sensed value back to Serial Port. There are only 4 possible resulting values from the sensor. Line is assumed to be black on white paper.

```
#include <MeMCore.h>

MeLineFollower lineFinder(PORT_2);

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int sensorState = lineFinder.readSensors();
  switch(sensorState)
  {
    case S1_IN_S2_IN:   Serial.println("S1_IN_S2_IN"); break;
    case S1_IN_S2_OUT:  Serial.println("S1_IN_S2_OUT"); break;
    case S1_OUT_S2_IN:  Serial.println("S1_OUT_S2_IN"); break;
    case S1_OUT_S2_OUT: Serial.println("S1_OUT_S2_OUT"); break;
    default: break;
  }
  delay(200);
}
```

## Input- Light Intensity Sensor

http://learn.makeblock.com/en/Makeblock-library-for-Arduino/class_me_light_sensor.html

Sample code below will read the light sensor, write the sensed value back to Serial Port. The measured value range from 0 (dimmest) to 1023 (brightest)

```
#include <MeMCore.h>

MeLightSensor lightSensor(PORT_8);

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.print("value = ");           // Print the results to the serial monitor
  Serial.println(lightSensor.read()); // Brightness value from 0-1023
  delay(50);                          // Wait 50 milliseconds before next measurement
}
```

## Input- Ultrasonic Distance Sensor

http://learn.makeblock.com/en/Makeblock-library-for-Arduino/class_me_ultrasonic_sensor.html

Sample code below will read the ultrasonic distance sensor, write the sensed value back to Serial Port.
The measured value range from 3cm to 400cm.
Closer than 3cm or farther than 400cm measurement will appear as 400cm, it is not possible to distinguish between the two.

```
#include <MeMCore.h>

MeUltrasonicSensor ultrasonic(PORT_3);

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.print("distance(cm) = ");          // Print the results to the serial monitor
  Serial.println(ultrasonic.distanceCm());  // Distance value from 3cm - 400cm
  delay(50);                                // Wait 50 milliseconds before next measurement
}
```

## Input- Button (On board)

Sample code below will read the onboard button's value and print different message to Serial Port depending on the button status.

```
#include <MeMCore.h>

void setup()
{
  Serial.begin(9600);
  pinMode(7, INPUT); //Define button pin as input
}

void loop() {
  if (analogRead(7) < 100){
  Serial.println("Button Pressed");
  }else{
  Serial.println("Not Pressed");
  }
  delay(50);
}
```

Sample code below contains a while loop to halt program until button is pressed.

```
#include <MeMCore.h>

void setup()
{
  //Wait until Onborad Button is pressed
  pinMode(7, INPUT); //Define button pin as input
  while (analogRead(7) > 100) { // While (Button is not pressed)
    delay(50);
  }
  //Other Code
}

void loop() {
  //Other Code
}
```

Sample code below will keep track of the button status and print a different message when the button is just-pressed and just-released.

```
#include <MeMCore.h>

void setup()
{
  Serial.begin(9600);
}

boolean currentPressed = false;
boolean pre_buttonPressed = false;

void loop() {
  currentPressed = (analogRead(7) < 100);
  if (currentPressed != pre_buttonPressed)
  {
    pre_buttonPressed = currentPressed;
    if (currentPressed == true)
    {
```

```
      Serial.println("Button Down (Pressed)");
    } else
    {
      Serial.println("Button Up (Released)");
    }
  }
}
```

## Input- Infrared Sensor from IR Remote

http://learn.makeblock.com/en/Makeblock-library-for-Arduino/class_me_i_r.html

Sample code below will constantly check for IR Input, when IR serial information is detected, the received code will be decoded using the codebook of Makeblock's IR remote. The original code and the pressed button are printed to Serial Port.
There are 21 buttons on the IR Remote.

```
#include <MeMCore.h>

MeIR ir;
void setup()
{
    ir.begin();
    Serial.begin(9600);
    Serial.println("Infrared Receiver Decoder");
}

void loop()
{
    if(ir.decode())
    {
      uint32_t value = ir.value;
      Serial.print("Raw Value: ");
      Serial.println(value);
      value = value >> 16 & 0xff;
      Serial.print("Button Code: ");
      Serial.println(value);
      Serial.print("Button: ");
       switch(value)
        {
            case IR_BUTTON_A: Serial.println("A");break;
            case IR_BUTTON_B: Serial.println("B");break;
            case IR_BUTTON_C: Serial.println("C");break;
            case IR_BUTTON_D: Serial.println("D");break;
            case IR_BUTTON_E: Serial.println("E");break;
            case IR_BUTTON_F: Serial.println("F");break;
            case IR_BUTTON_SETTING : Serial.println("Setting");break;
            case IR_BUTTON_LEFT: Serial.println("Left");break;
            case IR_BUTTON_RIGHT: Serial.println("Right");break;
            case IR_BUTTON_UP: Serial.println("Up");break;
            case IR_BUTTON_DOWN: Serial.println("Down");break;
            case IR_BUTTON_0: Serial.println("0");break;
            case IR_BUTTON_1: Serial.println("1");break;
            case IR_BUTTON_2: Serial.println("2");break;
            case IR_BUTTON_3: Serial.println("3");break;
```

```
            case IR_BUTTON_4: Serial.println("4");break;
            case IR_BUTTON_5: Serial.println("5");break;
            case IR_BUTTON_6: Serial.println("6");break;
            case IR_BUTTON_7: Serial.println("7");break;
            case IR_BUTTON_8: Serial.println("8");break;
            case IR_BUTTON_9: Serial.println("9");break;
            default:break;
        }
    }
}
```

## Example - Basic Line Following

Sample code below is a basic line following algorithm. Four possible states of the line sensor provides five different motor response.

| Left Sensor | Right Sensor | Sensor Reading | Motor Response | Left Motor Power | Right Motor Power |
|---|---|---|---|---|---|
| In | In | S1_IN_S2_IN | Go Straight | 255 | 255 |
| In | Out | S1_IN_S2_OUT | Left turn | 0 | 255 |
| Out | In | S1_OUT_S2_IN | Right turn | 255 | 0 |
| Out | Out | S1_OUT_S2_OUT | (If previously left turn) Left Turn | 0 | 255 |
| | | | (If previously right turn) Right Turn | 255 | 0 |

```
#include <MeMCore.h>

MeBuzzer buzzer;
MeLineFollower lineFinder(PORT_2);
MeDCMotor motor1(M1); //Motor1 is Left Motor
MeDCMotor motor2(M2); //Motor2 is Left Motor
MeRGBLed led(0, 30);

void setup() {
  led.setpin(13);
  pinMode(7, INPUT); //Define button pin as input
  while (analogRead(7) > 100) {
    delay(50); //Wait till button pressed to start.
  }
  buzzer.tone(200, 200);   //Buzzer beep to indicate start
}

float MOTOR1_TUNE = -1.0; //Left motor scale factor
```

```
float MOTOR2_TUNE = 1.0;  //Right motor scale factor
float turning_left = true;

void loop() {
  int sensorState = lineFinder.readSensors();
  switch (sensorState)
  {
    case S1_IN_S2_IN:
      motor1.run(MOTOR1_TUNE * 255.0); //Left motor Run
      motor2.run(MOTOR2_TUNE * 255.0); //Right motor Run
      led.setColorAt(1, 0, 255, 0); //Set LED1 (RGBLED2) (LeftSide)
      led.setColorAt(0, 0, 255, 0); //Set LED0 (RGBLED1) (RightSide)
      led.show();
      break;
    case S1_IN_S2_OUT:
      //turn left
      motor1.run(MOTOR1_TUNE * 0);      //Left motor Stop
      motor2.run(MOTOR2_TUNE * 255.0);  //Right motor Run
      led.setColorAt(1, 0, 0, 0);   //Set LED1 (RGBLED2) (LeftSide)
      led.setColorAt(0, 0, 255, 0); //Set LED0 (RGBLED1) (RightSide)
      led.show();
      turning_left = true;
      break;
    case S1_OUT_S2_IN:
      //turn right
      motor1.run(MOTOR1_TUNE * 255.0);  //Left motor Run
      motor2.run(MOTOR2_TUNE * 0);      //Right motor Stop
      led.setColorAt(1, 0, 255, 0); //Set LED1 (RGBLED2) (LeftSide)
      led.setColorAt(0, 0, 0, 0);   //Set LED0 (RGBLED1) (RightSide)
      led.show();
      turning_left = false;
      break;
    case S1_OUT_S2_OUT:
      //keep turning what it was turning
      if (turning_left) {
        motor1.run(MOTOR1_TUNE * 0);      //Left motor Stop
        motor2.run(MOTOR2_TUNE * 255.0);  //Right motor Run
        led.setColorAt(1, 0, 0, 0);   //Set LED1 (RGBLED2) (LeftSide)
        led.setColorAt(0, 255, 0, 0); //Set LED0 (RGBLED1) (RightSide)
        led.show();
      } else {
        motor1.run(MOTOR1_TUNE * 255.0);  //Left motor Run
        motor2.run(MOTOR2_TUNE * 0);      //Right motor Stop
        led.setColorAt(1, 255, 0, 0);   //Set LED1 (RGBLED2) (LeftSide)
        led.setColorAt(0, 0, 0, 0);     //Set LED0 (RGBLED1) (RightSide)
        led.show();
      }
      break;
    default: break;
  }
}
```

## Example - Ultrasonic Lap Timer (for timing MBot Race)

Example code below will use the ultrasonic sensor as an object detector for timing mBot lap race.

- Buzzer will provide audio feedback when object is detected.
- Elapsed Time and Lap Time in milliseconds will be printed to serial port.
- Detection distance (10cm) can be adjusted.
- Minimal lap time (1000ms) can be adjusted to avoid mistrigger.

```
#include <MeMCore.h>

MeBuzzer buzzer;
const int buzzerDuration = 20;

MeUltrasonicSensor ultrasonic(PORT_3);
const float distanceThreshold = 10.0;
float distance = 10.0;
boolean detected = false;
unsigned int detectCount = 0;

unsigned long currentTime = 0;
unsigned long firstDetectMills = 0;
unsigned long lastDetectMills = 0;
const float minimumLapTime = 1000; //ms

void setup()
{
  Serial.begin(115200);
  Serial.println("Lap Timer.");
  Serial.println("Trigger the sensor to start timing");
  Serial.print("Sensor detection distance: ");
  Serial.print(distance);
  Serial.println("cm");
  buzzer.tone(600, buzzerDuration);   //Buzzer sounds 600Hz for 1000ms
  delay(100);
  buzzer.tone(600, buzzerDuration);   //Buzzer sounds 600Hz for 1000ms
  delay(100);
  buzzer.tone(600, buzzerDuration);   //Buzzer sounds 600Hz for 1000ms
  delay(100);
  buzzer.tone(900, buzzerDuration * 2); //Buzzer sounds 600Hz for 1000ms
}

void loop()
{
  currentTime = millis();
  if ((currentTime - lastDetectMills) > minimumLapTime) {
    distance = ultrasonic.distanceCm();
    if ((distance < distanceThreshold)) {
      if (!detected) {
        detected = true;
        if (detectCount == 0) {
          firstDetectMills = currentTime;
          buzzer.tone(300, buzzerDuration);   //Buzzer sounds 600Hz for 1000ms
        }
        Serial.print("Lap:");
```

```
        Serial.print(detectCount);
        Serial.print(",  Time:");
        Serial.print(currentTime - firstDetectMills);
        Serial.print("ms,  LapTime:");
        Serial.print(currentTime - lastDetectMills);
        Serial.print(" ms, UltrasoundDistance:");
        Serial.print(distance);
        Serial.println(" cm");
        buzzer.tone(600, buzzerDuration);   //Buzzer sounds 600Hz for 1000ms
        detectCount++;
        lastDetectMills = currentTime;
      }
    } else {
      detected = false;
    }//distance < distanceThreshold
  }//minimumLapTime
}//loop
```

PID Proportional Integral Derivative Controller for DC Motor