## Setup:

1) Download and extract required files: **Download Link**

2) Navigate to the ServerCode folder and find the code for the type of Arduino you are using. (only Arduino Uno is supported so far). Open the .ino file in the Arduino IDE.
   **Download the Arduino IDE here.**
   Upload the code to your Arduino.

## Connecting to the Arduino from MATLAB:

1) All the required MATLAB code is contained in **/Client Code/MATLAB/Arduino.m**. Copy this file into your working directory.

2) Create a new Arduino object using the constructor: `Arduino(port)`, and connect using `Arduino.connect()`.

```
>> arduino = Arduino('COM7');
>> arduino.connect();
Connected to Arduino Uno running server code by Nigel Bess: https://github.com/NigelBess/Arduino-Server
```

If MATLAB did not print the above message, make sure the Arduino has the correct server code by following the instructions in the **Setup** Section.

## Arduino Functions:

Most functions require a pin parameter. This parameter can be either a string or an integer. Use integer for performance or string for readability.

| | String Form | Integer Form |
|---|---|---|
| **Digital Pin 0** | D0 | 0 |
| **Digital Pin 1** | D1 | 1 |
| **Digital Pin 2** | D2 | 2 |
| **Digital Pin 3** | D3 | 3 |
| **Digital Pin 4** | D4 | 4 |
| **Digital Pin 5** | D5 | 5 |

| | | |
|---|---|---|
| **Digital Pin 6** | D6 | 6 |
| **Digital Pin 7** | D7 | 7 |
| **Digital Pin 8** | D8 | 8 |
| **Digital Pin 9** | D9 | 9 |
| **Digital Pin 10** | D10 | 10 |
| **Digital Pin 11** | D11 | 11 |
| **Digital Pin 12** | D12 | 12 |
| **Digital Pin 13** | D13 | 13 |
| **Analog Pin 0** | A0 | 14 |
| **Analog Pin 1** | A1 | 15 |
| **Analog Pin 2** | A2 | 16 |
| **Analog Pin 3** | A3 | 17 |
| **Analog Pin 4** | A4 | 18 |
| **Analog Pin 5** | A5 | 19 |

**Arduino.pinMode(pin, type)**
Sets up [pin] for use as [type]. [type] can be either output, input or input with pullup, which can be defined as a string or integer. Use integer for performance or string for readability.

| | **String Form** | **Integer Form** |
|---|---|---|
| **Input** | INPUT | 0 |
| **Output** | OUTPUT | 1 |
| **Input with Pullup** | INPUT_PULLUP | 2 |

More information on pin types: https://www.arduino.cc/en/Tutorial/DigitalPins

**Arduino.digitalWrite(pin,state)**
Writes [state] to [pin]. [state] can be 0 (low) or 1 (high).
Requires [pin] to be set up using **Arduino.pinMode().**

**Arduino.analogWrite(pin,value)**
Writes [value] to [pin]. [value] is a voltage between 0 and 5v.
Requires [pin] to be set up using **Arduino.pinMode().**

> **Note:** Arduino UNO PWM output pins are D3, D5, D6, D9, D10 and D11. analogWrite will act as digitalWrite for other pins. Values greater than 2.5 will correspond to 1.

**Arduino.digitalRead(pinNumber)**
Returns the state of [pin], as 0 (low) or 1 (high).
Requires [pin] to be set up using **Arduino.pinMode().**

**Arduino.analogRead(pin)**
Returns the voltage at [pin].
Requires [pin] to be set up using **Arduino.pinMode().**

**Arduino.attachServo(pin)**
Sets up servo control at [pin].

**Arduino.writeServo(pin,angle)**
Writes angle (in degrees between 0 and 180) to servo at [pin].
Requires [pin] to be set up using **Arduino.attachServo().**

**Arduino.detachServo(pin)**
Detaches servo at [pin], and frees up [pin] for other use.

**Arduino.encoder(interruptPin,[optional]secondaryPin)**
Sets up an encoder with channel A at [interruptPin] (must be D2 or D3), and channel B at [secondaryPin] (any other pin). Omitting [secondaryPin] will cause the encoder to behave as an incremental encoder, as opposed to a quadrature encoder.

**Arduino.readEncoder(interruptPin)**
Returns the count of the encoder at [interruptPin].
Requires [interruptPin] to be set up using **Arduino.encoder().**

**Arduino.resetEncoder(interruptPin)**
Sets the count of encoder at [interruptPin] to zero.
Requires [interruptPin] to be set up using **Arduino.encoder().**

**Arduino.setEncoderDirection(interruptPin,direction)**

Sets the direction in which encoder ticks are counted, depending on the value of the direction parameter.

| Direction Value | Effect |
|---|---|
| 0 | Do not count ticks |
| 1 | Count ticks with normal polarity |
| 2 | Count ticks with opposite polarity |

**Arduino.detachEncoder(interruptPin)**
Detaches the encoder at interruptPin, and frees up [interruptPin] for other use.

**Arduino.checkConnection()**
Used to make sure that the arduino is still connected. It doesn't make the arduino do anything, but will throw an error if the arduino disconnected.