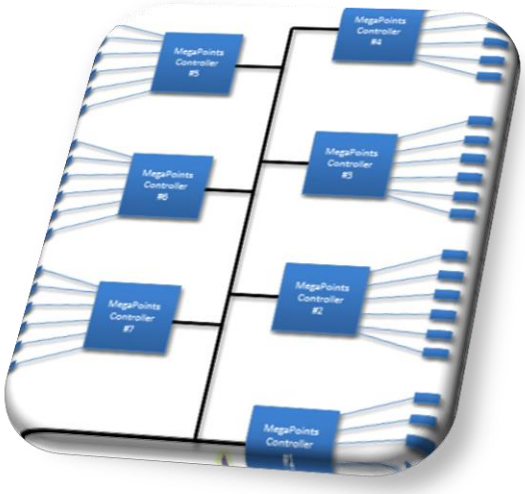
MegaPoints hacking Guide

Interfacing MegaPoints Controllers with an Arduino or Raspberry Pi.



Hacking guide

Revision 5 January 2016

Email: info@megapointscontrollers.com

Contents

Introduction

Ever fancied hooking up to an Arduino or Raspberry Pi and just having a go? This document will get you started controlling MegaPoints Controllers products using these devices. This guide will build up over time to include all products on both the Pi and Arduino.

Please understand this before starting:

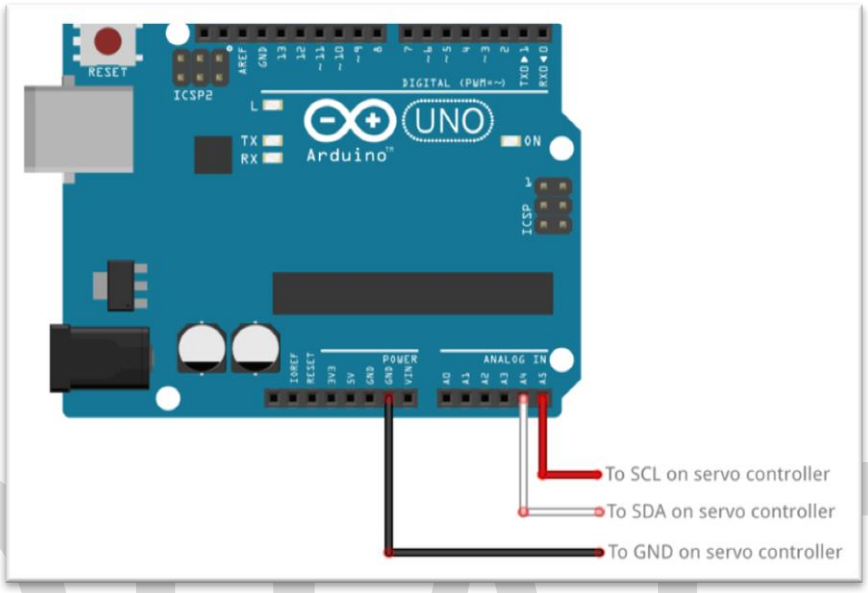
> *This guide is provided as is and totally unsupported. Please do not ask for code troubleshooting or to coding advice. We provide a range of products that are guaranteed to work when used together. Adding in your devices may change the desired behaviour.*

We don't cover setting up the build environment for the Arduino or Raspberry Pi, though do include package details where necessary for execution of the code.

Check the video section for examples of running code.

Servo Controller

Connect the servo controller network port (either will do) to an Arduino Uno in the following manner:



Set your Servo Controller to slave mode. Set the address to 2 (see Servo Controller documentation for further details).

The Servo Controller protocol is very network efficient and requires only two bytes to operate.

	Byte 1								Byte 2							
Servo	8	7	6	5	4	3	2	1	X	X	X	X	9	10	11	12
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

One only needs to send two bytes to the Servo Controller address to observe movement.

Arduino

After setting the servo controller address to 2, compile the following code:

```
#include <Wire.h>

const byte ServoControllerAddr = 2;     // Set the servo controller address

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

void loop(){
  // Move all servos to LOW position and wait 5 seconds.
  Serial.println("Moving all servos to LOW position.");
  Wire.beginTransmission(ServoControllerAddr);
  Wire.write(0);
  Wire.write(0);
  Wire.endTransmission();
  delay(5000);
```

```
  // Move all servos to random positions and wait 5 seconds.
  Serial.println("Moving all servos to random positions.");
  Wire.beginTransmission(ServoControllerAddr);
  Wire.write(random(0,255));
  Wire.write(random(0,255));
  Wire.endTransmission();
  delay(5000);

  // Move all servos to high position and wait 5 seconds.
  Serial.println("Moving all servos to HIGH position.");
  Wire.beginTransmission(ServoControllerAddr);
  Wire.write(255);
  Wire.write(15);
  Wire.endTransmission();
  delay(5000);
}
```

Raspberry Pi

I started with a fresh installation of the Raspbian operating system. For details on how to install see https://www.raspberrypi.org. It is assumed you have a Raspberry Pi up and running. I'm using a model B, however this should work on all models. Please understand the following:

No warranty is implied or offered. This is completely unsupported.

Edit the file sudo su –

and comment out the i2c line thus:

```
#blacklist i2c-bcm2708
```

Edit /etc/modules and add the I2C module:

```
i2c-dev
```

Download the required packages:

```
sudo apt-get update
sudo apt-get install i2c-tools
sudo apt-get install python-smbus
```

Reboot the Pi

```
sudo reboot
```

Python script

```
import smbus
bus = smbus.SMBus(1)
bus.write_i2c_block_data(0x02,0x00,[255,255]
bus.write_i2c_block_data(0x02,0x00,[0,0]
```

The above section in orange is incomplete and not yet working.

DCC Module

Arduino

Raspberry Pi

MultiPanel

There are two examples for interfacing the MultiPanel, one for transmitting data and another for receiving.

To address up to 192 servos on the MegaPoints Controllers network the MultiPanel divides the 192 servo addresses into 24 banks of 8 bits each. Therefore to address any particular servo one needs to know the bank number and then set the bits within that bank. The following table illustrates this:

Servo Controller Network Address	Servo Range	MultiPanel Bank
2	1 - 8	0
2,3	9 - 16	1
3	17 – 24	2
4	25 – 32	3
4,5	33 – 40	4
5	41 – 48	5
6	49 – 56	6
6,7	57 – 64	7
7	65 – 72	8
8	73 – 80	9
8,9	81 – 88	10
9	89 – 96	11
10	97 – 104	12
10,11	105 – 112	13
11	113 – 120	14
12	121 – 128	15
12,13	129 – 136	16
13	137 – 144	17
14	145 – 152	18
14,15	153 – 160	19
15	161 – 168	20
16	169 – 176	21
16,17	177 – 184	22
17	185 – 192	23

The MultiPanel Master will transmit data for each bank according to its internal scheduler. When idle the next bank is transmitted at 100 MS intervals. If a button is pressed the appropriate bank is immediately transmitted. This means that data can arrive in any order but is always prefixed by the bank number.

Arduino

Read Data

This code demonstrates how to read MultiPanel Data that is transmitted by the internal scheduler. The data received will be displayed in the serial monitor.

```
/*

        MegaPoints Controllers MultiPanel Master read usage example
        No warranty is implied or offered.
        This is unsupported.
        Web: megapointscontrollers.com <-- Home of the MegaPoints Controllers system.

        The MegaPoints Controllers network will control up to 192 servos via 16 Servo
Controllers. To address
        servos in the range 0 - 191 each MultiPanel has 24 banks of one byte. By specifying
the bank number (0-23)
        and setting the individual bits a servo can be moved on or off in accordance with it's
Servo Controller local programming.

        This example sets up a service routine that is called when I2C data is received.
        The bank address and data are displayed on the serial port.

*/

#include <Wire.h>                    // We need the wire library
#define SlaveAddr 21                   // Set out I2C address to 21 so we can receive updates
from the MultiPanel master.
uint8_t MultiPanelBank = 0;          // Bank address (0-23)
uint8_t MultiPanelData = 0;          // Data
uint8_t MultiPanelDataReady = 0;     // Fresh data indicator

void setup(){
  Serial.begin(115200);             // Serial port to 115200. Adjust as required.
  Wire.begin(SlaveAddr);             // Initialise I2C comms, dont care what our address is as
were only transmitting data.
  Serial.println("Listening for MegaPoints Controllers MultiPanel data ...");
  Serial.println("\tBank\tData");
  Wire.onReceive(GetData);          // Set up the interrupt data handler routine.
}


void loop(){
  if (MultiPanelDataReady == 1){      // Check if we have data, if so, continue.
    Serial.print("\t");
    Serial.print(MultiPanelBank);     // Print the bank number.
    Serial.print("\t");
    Serial.print(MultiPanelData);     // Print the data.
    Serial.println();
    MultiPanelDataReady = 0;          // reset data indicator back to 0.
  }
}

void GetData(int howmany){            // This function is called when data is received.
  MultiPanelBank = Wire.read();       // Read the bank number (0-23).
  MultiPanelData = Wire.read();       // Read the data byte.
  MultiPanelDataReady = 1;            // Set the data ready flag.
}
```

Write Data

This code demonstrates how to write data to a specific MultiPanel bank.

```
/*

        MegaPoints Controllers MultiPanel Master write usage example
        No warranty is implied or offered.
        This is unsupported.
        Web: megapointscontrollers.com <-- Home of the MegaPoints Controllers system.

       The MegaPoints Controllers network will control up to 192 servos via 16 Servo
Controllers. To address
       servos in the range 0 – 191 each MultiPanel has 24 banks of one byte. By specifying the
bank number (0-23)
       and setting the individual bits a servo can be moved on or off in accordance with it's
Servo Controller local programming.

       When communicating with a MultiPanel Master two bytes must be sent, bank number (0-23)
and data byte (0-255).

        For example:
        To turn servos 1-8 OFF you would send the data 0 & 0.
          Wire.write(0);
          Wire.write(0);

        To turn servos 1 & 8 ON and 2-7 OFF you would send 0 & B10000001.
          Wire.write(0);
          Wire.write(B10000001);

        To turn on servo 12 you would send 1 & B0001000.
          Wire.write(0);
          Wire.write(B0010000);

        To turn on servo 13 you would send 1 & B00000001.
          Wire.write(1);
          Wire.write(B00000001);

        Don't forget that each time the state machine is updated for any servo it is locked
out for three seconds. Subsequent commands to this servo channel will be ignored during the
lockout period.

*/

#include <Wire.h>          // We need the wire library
#define MasterAddr 20

void setup(){
  Wire.begin();            // Initialise I2C comms, dont care what our address is as were only
transmitting data.
}


void loop(){
  for (int x = 0; x < 256; x++){
    Wire.beginTransmission(MasterAddr);      // Send to MultiPanel Master.
    Wire.write(0);                           // Select bank 0 (servos 1-8)
    Wire.write(x);                           // Count up from 0 ...
    Wire.endTransmission();                  // Dont forget to stop the transmission.

    Wire.beginTransmission(MasterAddr);
    Wire.write(2);                           // Select bank 2 (servos 17-24)
    Wire.write(x ^ 255);                     // Write the opposite of bank 0
    Wire.endTransmission();

    delay(3000);                             // Wait three seconds before looping to allow the
state machine lockout to clear.
  }
```

```
}
```

The serial monitor will look something like this:

```
Listening for MegaPoints Controllers MultiPanel data ...
        Bank    Data
        0       0
        1       0
        2       0
        3       255
        4       255
        5       255
        6       255
        7       255
        8       255
        9       255
        10      255
        11      255
        12      255
        13      255
        14      255
        15      255
        16      255
        17      255
        18      255
        19      255
        20      255
        21      255
        22      255
        23      255
        0       0
        1       0
        2       0
```

Raspberry Pi

Route Processor

Arduino

Raspberry Pi

Servo Controller

Arduino

Raspberry Pi

DCC Module

Arduino

Raspberry Pi

MultiPanel

Arduino

Raspberry Pi

Route Processor

Arduino

Raspberry Pi

Contacting us

Web: www.loolee.org
Email: mp@loolee.org
Phone: (+44) 07846 409320

If you have any product improvement suggestions we'd be very pleased to hear from you.

NOTE: We operate on a policy of continuous improvement. Colours, specifications and even the placement of components may vary from time to time. Documentation will continue to be updated to reflect changes or answer frequent customer questions as they arise.