I2C Programming

https://www.jetsonhacks.com/2019/07/22/jetson-nano-using-i2c/

Step 1. Hardware Design. Connect the I2C ports.

In the default Jetson Nano Image, there are two I2C ports available on the J41 Header. From the Jetson Nano J41 Pinout :

I2C Bus 1 SDA is on Pin 3
I2C Bus 1 SCL is on Pin 5
I2C Bus 0 SDA is on Pin 27
I2C Bus 0 SCL is on Pin 28

Note: Before wiring the Jetson, make sure that the power is disconnected. When the power is plugged in, the power and ground rails on the headers are always live, even if the processor itself is off.

For the first demo in the video, we wire Bus 1:

J41 Pin 3 (SDA) -> PCA9685 SDA
J41 Pin 5 (SCL) -> PCA9685 SCL
J41 Pin 1 (3.3V) -> PCA9685 VCC
J41 Pin 6 (GND) -> PCA9685 GND

A 5V 4A power supply is connected to the PCA 9685. The SG90 micro server is connected to port 0 of the PWM outputs. Note that the GND signal is towards the outside edge of the board, the control signal is towards the center of the board.

After wiring the board, plug the Jetson in. Once the Nano is up and running, open a terminal and execute:

```
$ i2cdetect -y -r 1
```

The default address of the PCA9685 is 0x40 (this is hexadecimal 40). You should see an entry of ’40’ in the addresses listed. If you do not see the entry, then the wiring is probably incorrect. When the address does not show up, then you will not be able to use the device. Note: You can change the default address of the PCA9685, so you will need to take that into account when you check the device visibility.

Step 2. Software Testing Code

We are now ready to run our first demo:

```
$ python3
>>> from adafruit_servokit import ServoKit
>>> kit = ServoKit(channels=16)
>>> kit.servo[0].angle=137
>>> kit.servo[0].angle=25
>>> quit()
```

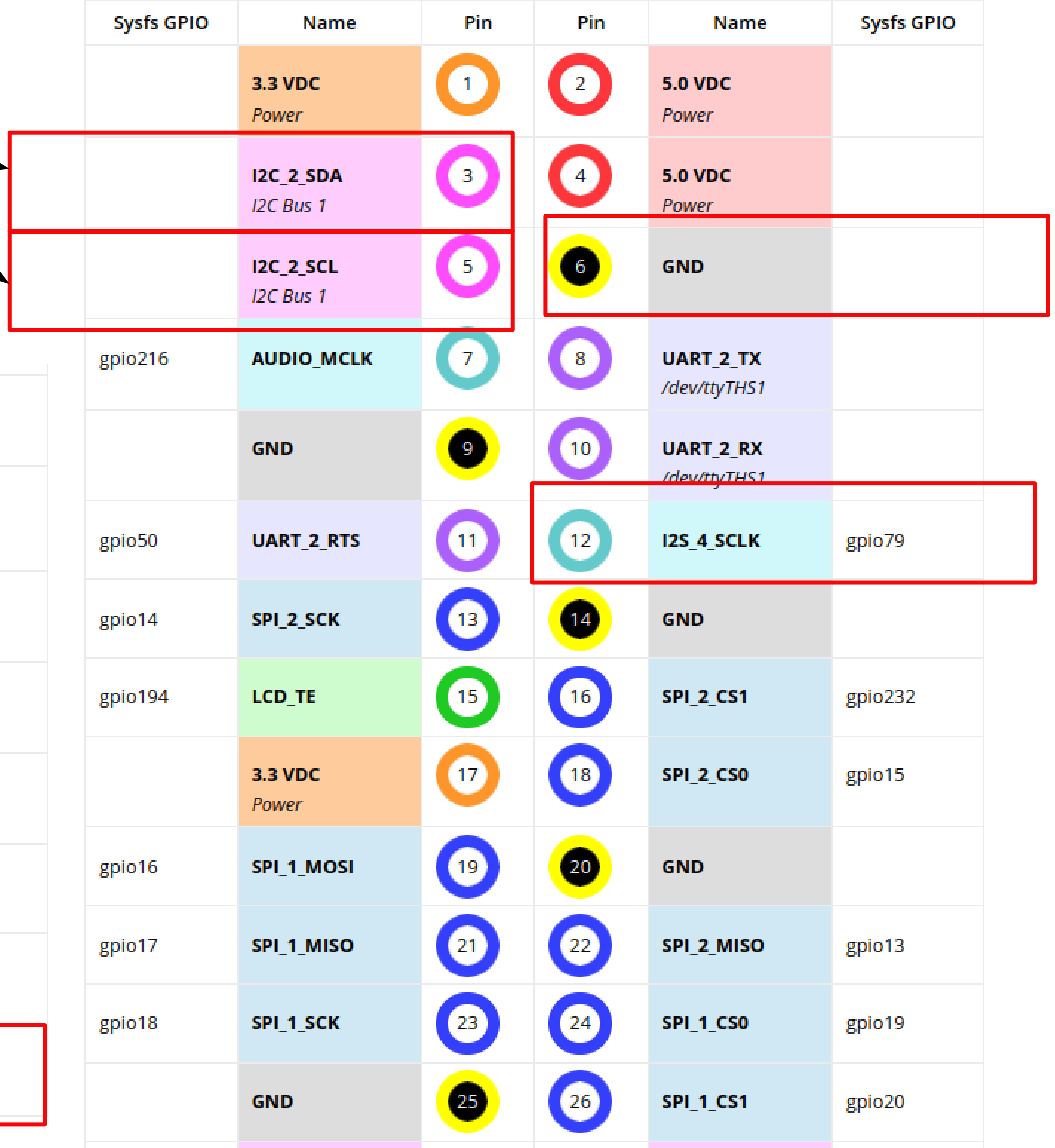
Our servo should move to the appropriate angle when we command it.

I2C on Jetson Nano J41 Header

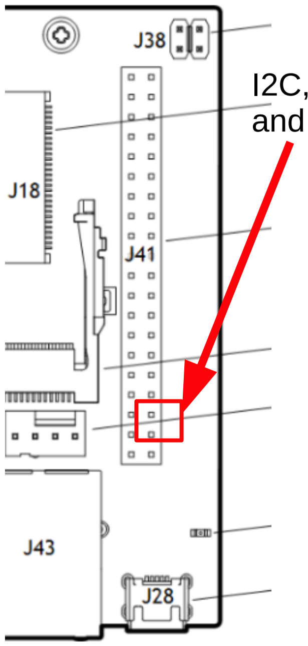
https://www.jetsonhacks.com/nvidia-jetson-nano-j41-header-pinout/

Note: I2C and UART pins are connected to hardware and should not be reassigned. By default, all other pins (except power) are assigned as GPIO. Pins labeled with other functions are recommended functions if using a different device tree.

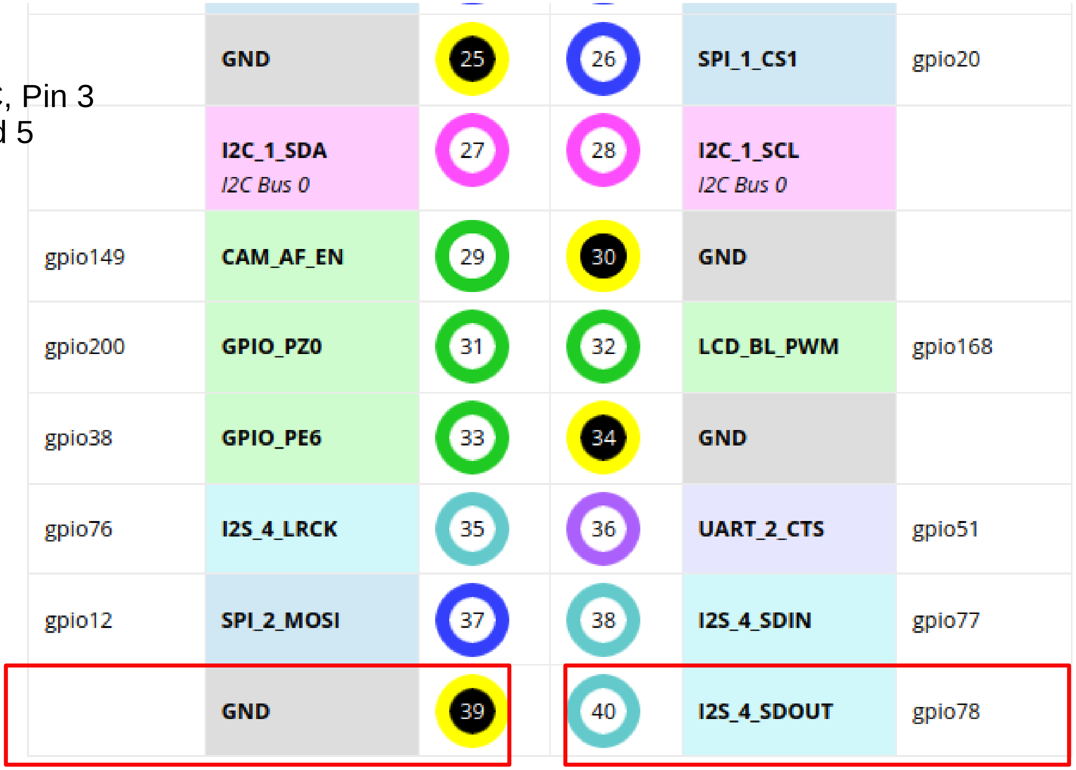
I2C, Pin 3 and 5 can be utilized for I2C interface design.



Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO
	3.3 VDC *Power*	1	2	**5.0 VDC** *Power*	
	I2C_2_SDA *I2C Bus 1*	3	4	**5.0 VDC** *Power*	
	I2C_2_SCL *I2C Bus 1*	5	6	**GND**	
gpio216	**AUDIO_MCLK**	7	8	**UART_2_TX** */dev/ttyTHS1*	
	GND	9	10	**UART_2_RX** */dev/ttyTHS1*	
gpio50	**UART_2_RTS**	11	12	**I2S_4_SCLK**	gpio79
gpio14	**SPI_2_SCK**	13	14	**GND**	
gpio194	**LCD_TE**	15	16	**SPI_2_CS1**	gpio232
	3.3 VDC *Power*	17	18	**SPI_2_CS0**	gpio15
gpio16	**SPI_1_MOSI**	19	20	**GND**	
gpio17	**SPI_1_MISO**	21	22	**SPI_2_MISO**	gpio13
gpio18	**SPI_1_SCK**	23	24	**SPI_1_CS0**	gpio19
	GND	25	26	**SPI_1_CS1**	gpio20



I2C, Pin 3 and 5



	GND	25	26	**SPI_1_CS1**	gpio20
	I2C_1_SDA *I2C Bus 0*	27	28	**I2C_1_SCL** *I2C Bus 0*	
gpio149	**CAM_AF_EN**	29	30	**GND**	
gpio200	**GPIO_PZ0**	31	32	**LCD_BL_PWM**	gpio168
gpio38	**GPIO_PE6**	33	34	**GND**	
gpio76	**I2S_4_LRCK**	35	36	**UART_2_CTS**	gpio51
gpio12	**SPI_2_MOSI**	37	38	**I2S_4_SDIN**	gpio77
	GND	39	40	**I2S_4_SDOUT**	gpio78

Harry Li, Ph.D.

Cross Reference: I2C Sample Design

Bill of Material

1. Adafruit Mini Pan-Tilt Kit - Assembled with Micro Servos, Figure 1.
2. SunFounder PCA9685 16 Channel 12 Bit PWM Servo Driver, Figure 2.
3. Micro servo, Figure 3
4. Dome camera enclosure, Figure 5.

Software Github
4. server kit github code with installation guide, Figure 4.
https://github.com/JetsonHacksNano/ServoKit
Reference: libi2c, https://github.com/amaork/libi2c

https://www.jetsonhacks.com/2019/07/22/jetson-nano-using-i2c/

```
$ git clone https://github.com/JetsonHacksNano/ServoKit
$ cd ServoKit
$ ./installServoKit.sh
```

This installs the ServoKit library and also sets up the I2C and GPIO permissions so that we can run programs from user space. GPIO permissions are added to support the underlying Jetson.GPIO library

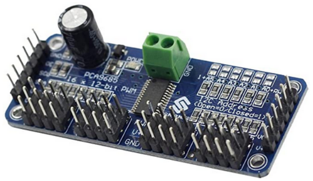
Figure 1. Adafruit Mini Pan-Tilt Kit - Assembled with Micro Servos $23.98
https://www.amazon.com/gp/product/B00PY3LQ2Y/ref=ox_sc_act_image_2?smid=AM0JQO74J587C&psc=1

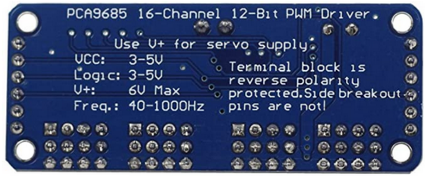
Figure 3. ElectroBot 2X Pcs Sg90 Micro Servo Motor 9G Rc Robot Helicopter Airplane Boat Controls

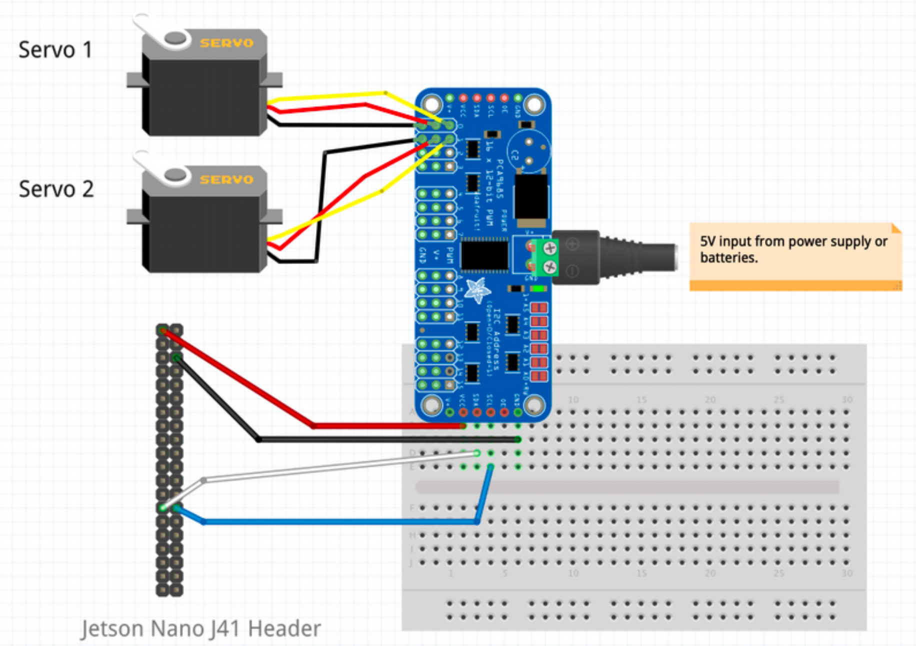
https://www.amazon.com/s?k=Electrobot&ref=bl_dp_s_web_20549721011

Figure 4. github code for i2c based servo drive
https://github.com/JetsonHacksNano/ServoKit

Figure 2. SunFounder PCA9685 16 Channel 12 Bit PWM Servo Driver for Arduino and Raspberry Pi
https://www.amazon.com/SunFounder-PCA9685-Channel-Arduino-Raspberry/dp/B014KTSMLA/ref=as_li_ss_tl?keywords=pca9685&qid=1563204174&s=gateway&sr=8-7&linkCode=sl1&tag=jetsonhacks-20&linkId=e40c8e444497217fd3a105e993e40388&language=en_US




PCA9685 16-Channel 12-Bit PWM Driver
Use V+ for servo supply
VCC: 3-5V
Logic: 3-5V
V+: 6V Max
Freq.: 40-1000Hz
Terminal block is reverse polarity protected. Side breakout pins are not!



Cross Reference: I2C Servo github Code Demo

https://www.jetsonhacks.com/2019/07/22/jetson-nano-using-i2c/

Demo 1

In the default Jetson Nano Image, there are two I2C ports available on the J41 Header. From the Jetson Nano J41 Pinout :

I2C Bus 1 SDA is on Pin 3
I2C Bus 1 SCL is on Pin 5
I2C Bus 0 SDA is on Pin 27
I2C Bus 0 SCL is on Pin 28

Note: Before wiring the Jetson, make sure that the power is disconnected. When the power is plugged in, the power and ground rails on the headers are always live, even if the processor itself is off.

For the first demo in the video, we wire Bus 1:

J41 Pin 3 (SDA) -> PCA9685 SDA
J41 Pin 5 (SCL) -> PCA9685 SCL
J41 Pin 1 (3.3V) -> PCA9685 VCC
J41 Pin 6 (GND) -> PCA9685 GND

A 5V 4A power supply is connected to the PCA 9685. The SG90 micro server is connected to port 0 of the PWM outputs. Note that the GND signal is towards the outside edge of the board, the control signal is towards the center of the board.

After wiring the board, plug the Jetson in. Once the Nano is up and running, open a terminal and execute:

```
$ i2cdetect -y -r 1
```

The default address of the PCA9685 is 0x40 (this is hexadecimal 40). You should see an entry of ’40’ in the addresses listed. If you do not see the entry, then the wiring is probably incorrect. When the address does not show up, then you will not be able to use the device. Note: You can change the default address of the PCA9685, so you will need to take that into account when you check the device visibility.

We are now ready to run our first demo:

```
$ python3
>>> from adafruit_servokit import ServoKit
>>> kit = ServoKit(channels=16)
>>> kit.servo[0].angle=137
>>> kit.servo[0].angle=25
>>> quit()
```

Our servo should move to the appropriate angle when we command it.

Cross Reference (Not Tested): libi2c github Code

To use:

1. Installation

2. Interface

3. Data structure

```
i2c_ioctl_write (once max 16 bytes) are more efficient than i2c_write (once max 4 bytes).
/* Close i2c bus */
void i2c_close(int bus);
/* Open i2c bus, return i2c bus fd */
int i2c_open(const char *bus_name);
/* I2C file I/O read, write */
ssize_t i2c_read(const I2CDevice *device, unsigned int iaddr, void *buf, size_t len);
ssize_t i2c_write(const I2CDevice *device, unsigned int iaddr, const void *buf, size_t len);
/* I2c ioctl read, write can set i2c flags */
ssize_t i2c_ioctl_read(const I2CDevice *device, unsigned int iaddr, void *buf, size_t len);
ssize_t i2c_ioctl_write(const I2CDevice *device, unsigned int iaddr, const void *buf, size_t len);
```