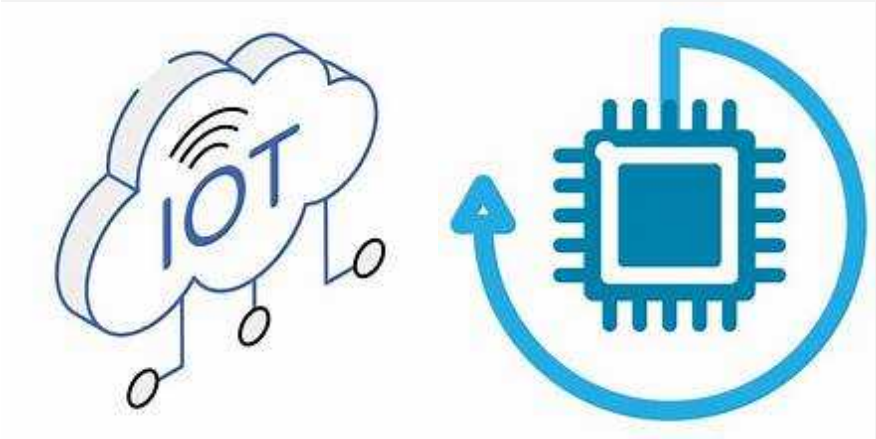
한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

Internet of Things (IoT) Systems



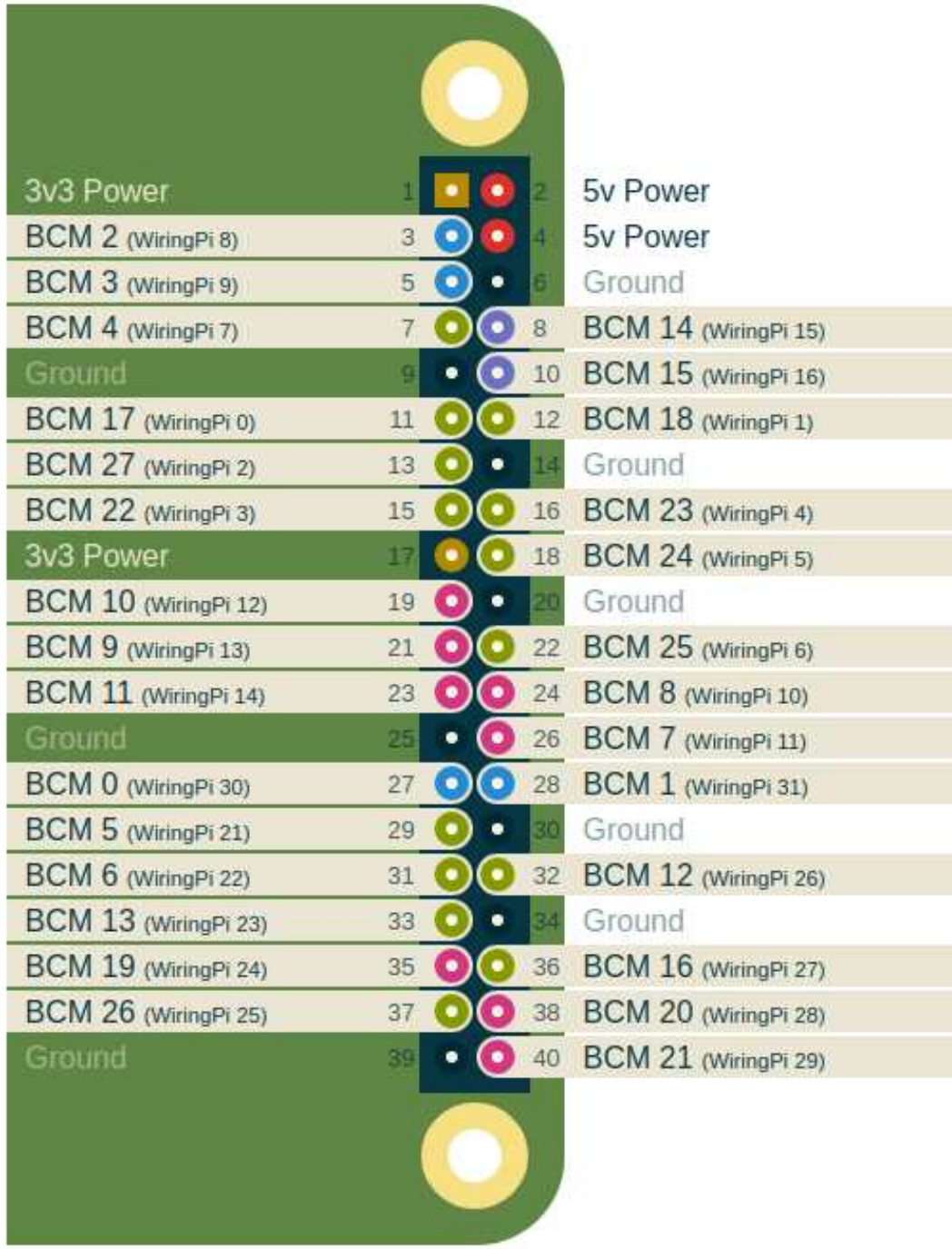
Week 6

Raspberry Pi Programming

Ikram Syed, Ph.D.
Associate Professor
Department of Information and Communication Engineering
Hankuk University of Foreign Studies (HUFS)

Spring – 2025

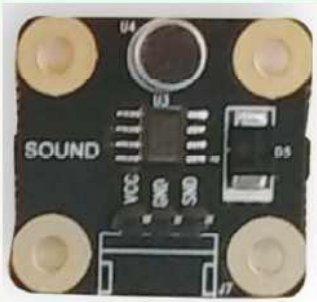
- **GPIO Pin Numbering Schemes**
  - Physical
    - The actual pin numbers on 40-pin connector
  - BCM
    - Broadcom pin numbers often called GPIO numbers
    - This is the most common method of naming the GPIO pins
  - WiringPi
    - Pin numbers used in WiringPi library



Access and Control of IoT Devices

- Sound Sensor detects surrounding sound
  - A microphone connected to sensor detects sound
  - This sensor can not measure magnitude or frequency of sound

<Table 3-5> Specifications of sound sensor

Shape	Category	Description
	Sound Sensor	Microphone
	Operating Voltage	5V
	I/O Interface	1 Analog OUTPUT
Sensor to detect sound		

Access and Control of IoT Devices

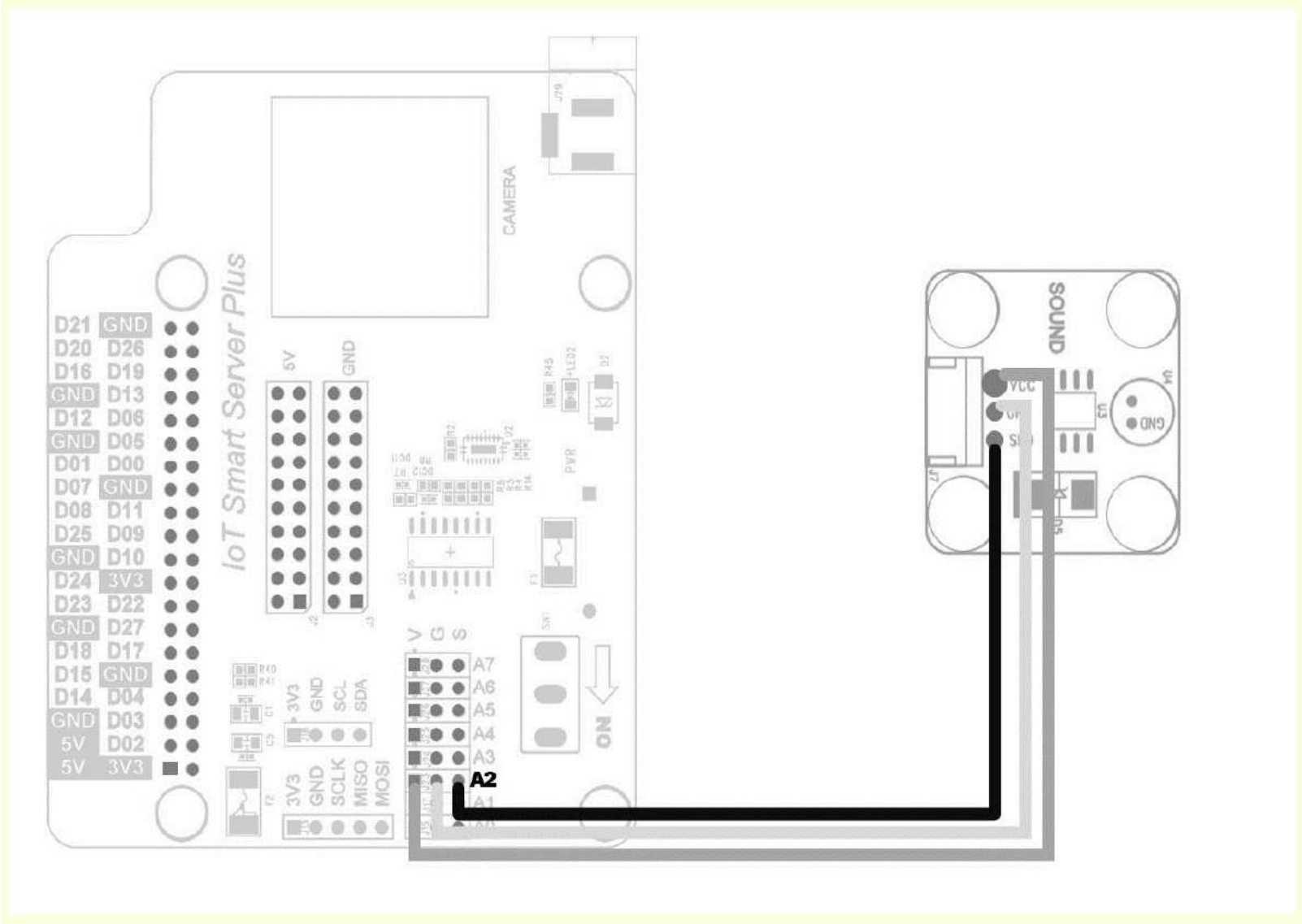
- Connecting Sound Sensor and RPi
  - **RPi does not have a pin to support ADC function**
  - Read value of analog sensor through external ADC chip MCP3208
  - Use SPI to control the chip

<Table 3-6> Pin Connection Information for SPI ADC and Sound Detection Sensor

ADC Port	Sound Module Pin No.
ADC2	SND

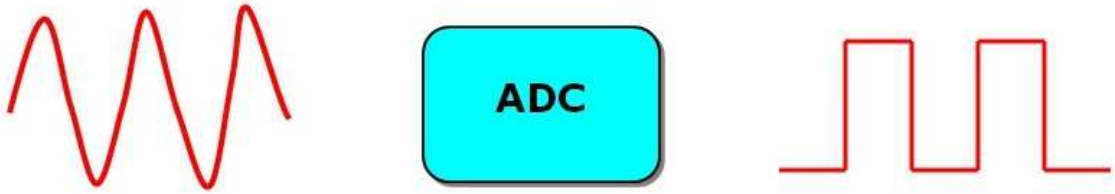
Connect SPI ADC port 2 with SND pin of sound sensor module through a cable

Connect module without applying power to RPi



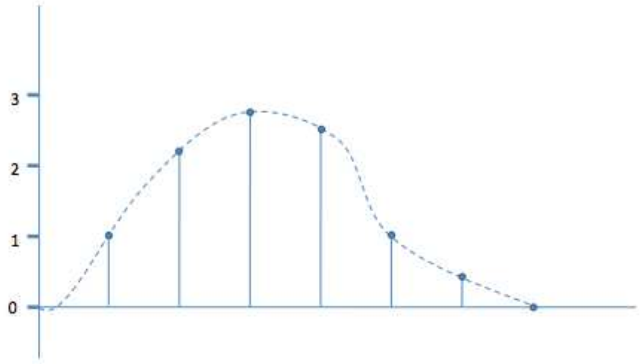
Access and Control of IoT Devices

- MCP3208 is a chip that converts analog signals to digital signals

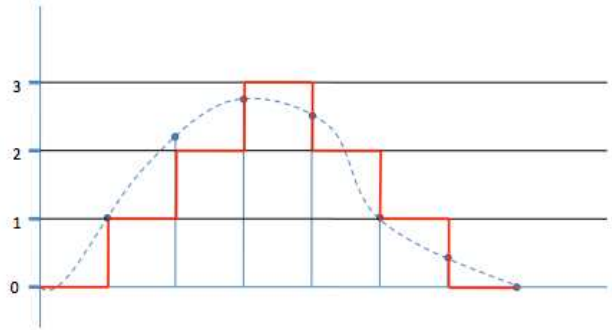


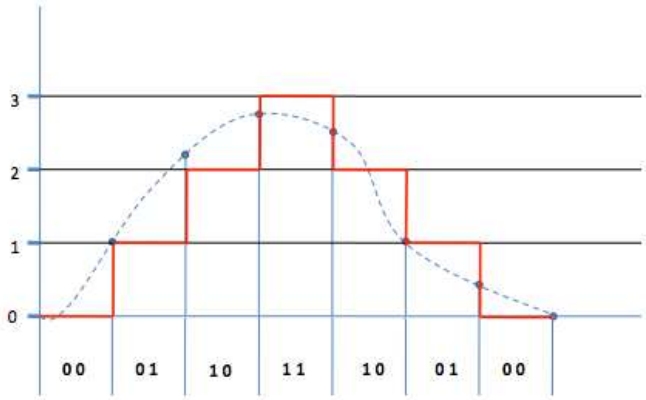
A → D Conversion Process

- Pulse Code Modulation
  - Sampling is the process of reading the values of the analog signal at discrete time intervals
  - Quantizing is the process of assigning a discrete value from a range of possible values to each sample obtained
    - The number of possible values will depend on the number of bits used to represent each sample
    - Quantization can be achieved by either rounding the signal up or down to the nearest available value
  - Encoding is the process of representing the sampled values as a binary number in range 0 to n



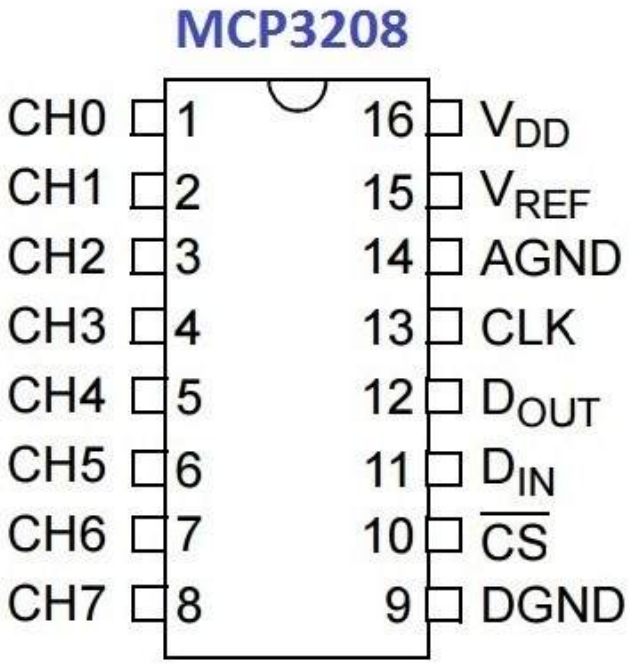
The signal is first sampled





Access and Control of IoT Devices

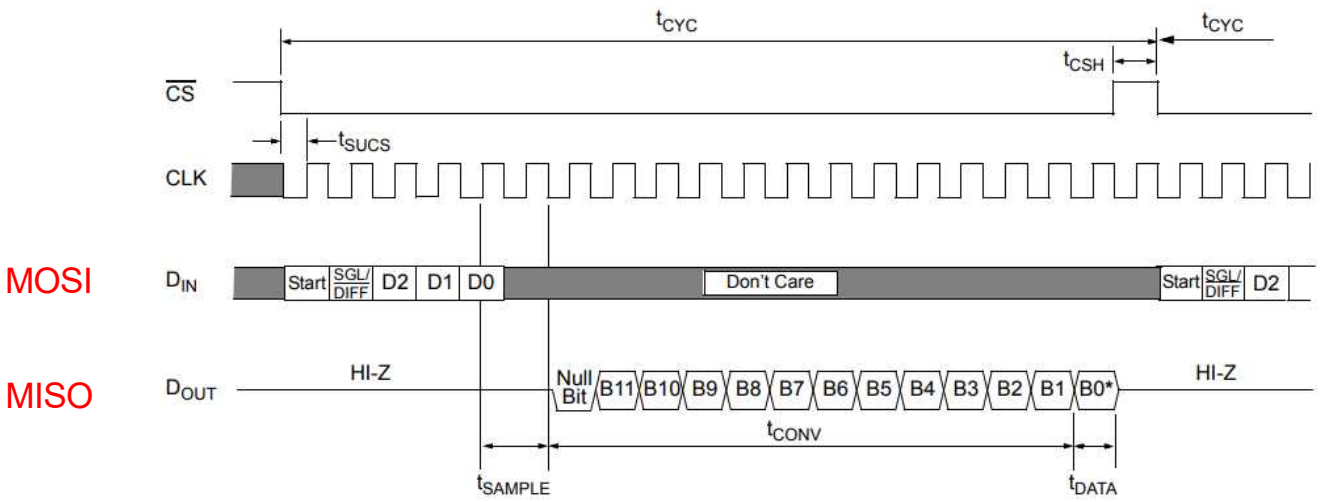
- MCP3208 supports 8 channels and 12 bit resolution



MCP3208	Signal	Description
1	CH0	Analogue Input 1
2	CH2	Analogue Input 2
3	CH2	Analogue Input 3
4	CH3	Analogue Input 4
5	CH4	Analogue Input 5
6	CH5	Analogue Input 6
7	CH6	Analogue Input 7
8	CH7	Analogue Input 8
-	NC	No Connection
-	NC	No Connection
9	DGND	Digital Ground
10	CS/SHDN	Chip Select/Shut Down
11	DIN	Digital Serial Input MOSI
12	DOUT	Digital Serial Output MISO
13	CLK	Clock
14	AGND	Analogue Input Ground
15	VREF	Reference Voltage
16	VDD	Positive Supply Voltage

Access and Control of IoT Devices

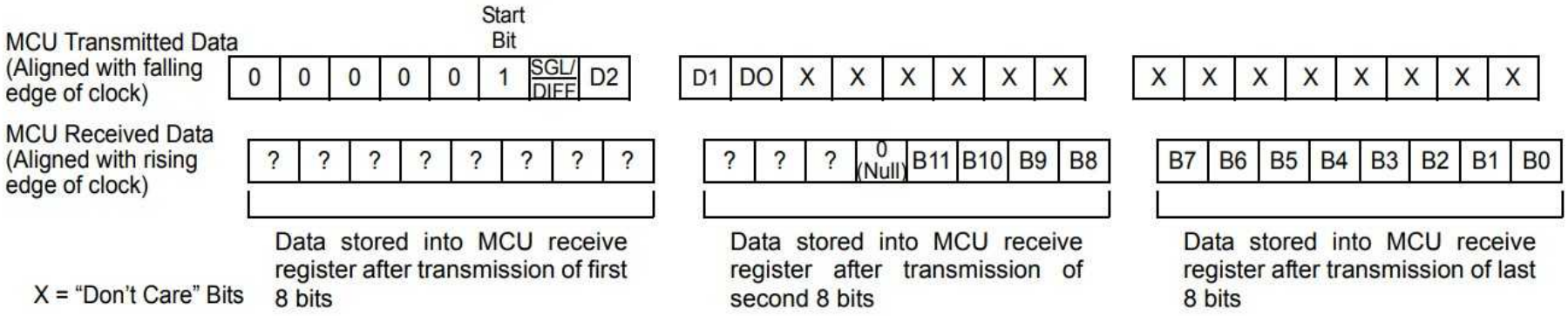
- Communicating with the MCP3208
  - To control chip, CS pin must be changed to LOW from HIGH
  - RPi must identify which channel (0–7) it wants to read
    - The channel is selected using a 4-bit identifier
  - MCP3208 then sends 12 bits of data back to the RPi

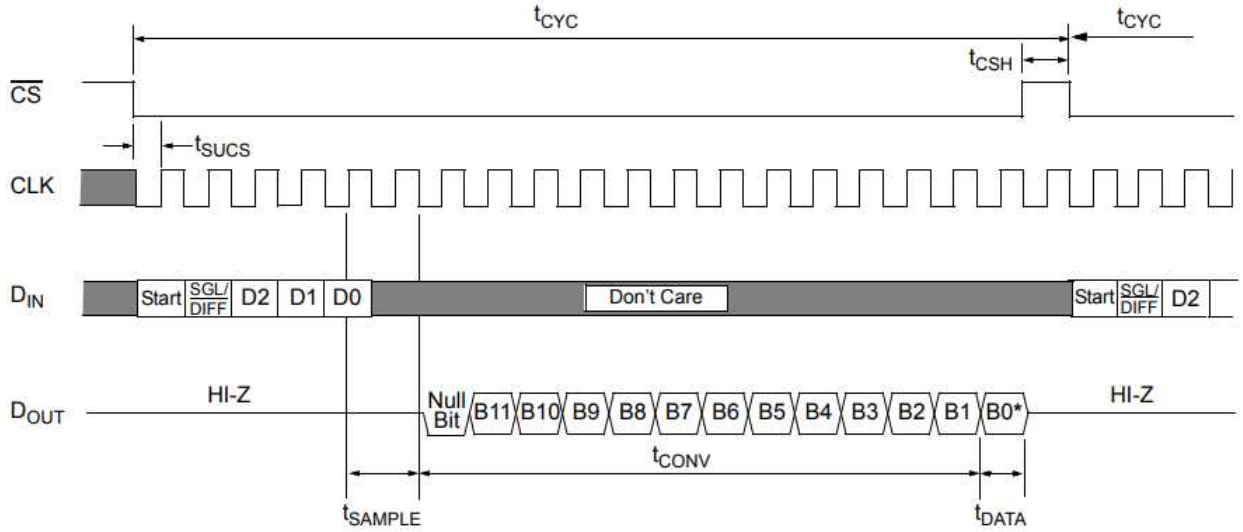


CONFIGURATION BITS FOR THE MCP3208

Control Bit Selections				Input Configuration	Channel Selection
Single /Diff	D2	D1	D0		
1	0	0	0	single-ended	CH0
1	0	0	1	single-ended	CH1
1	0	1	0	single-ended	CH2
1	0	1	1	single-ended	CH3
1	1	0	0	single-ended	CH4
1	1	0	1	single-ended	CH5
1	1	1	0	single-ended	CH6
1	1	1	1	single-ended	CH7

Access and Control of IoT Devices

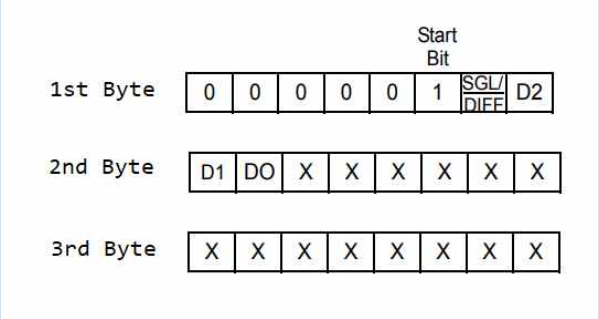


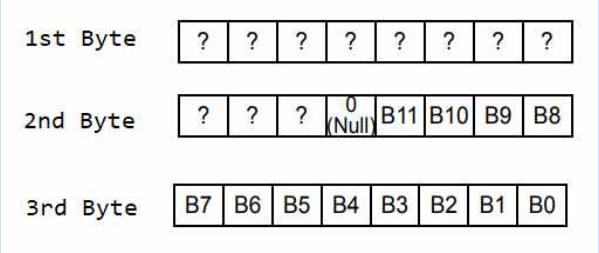


```c
#include <stdio.h>
#include <wiringPi.h>

#define SPI_CH 0
#define ADC_CH 2
#define ADC_CS 29
#define SPI_SPEED 500000

int main(void){
        int value=0, i;
        unsigned char buf[3];

        if(wiringPiSetup() == -1) return 1;

        if(wiringPiSPISetup() == -1) return -1;

        pinMode(ADC_CS,OUTPUT);

        for(i=0; i<20; i++){
                buf[0] = 0x06 | ((ADC_CH & 0x04)>>2);
                buf[1] = ((ADC_CH & 0x03)<<6);
                buf[2] = 0x00;

                digitalWrite(ADC_CS,0);

                wiringPiSPIDataRW(SPI_CH,buf,3);

                buf[1]=0x0F & buf[1];

                value = (buf[1]<<8) | buf[2];

                digitalWrite(ADC_CS,1);

                printf("%d\n",value);

                delay(100);
        }
}
```

wiringPiSPIDataRW performs a simultaneous write-read transaction over the selected SPI bus.

Data that was in your buffer is overwritten by data returned from the SPI bus.

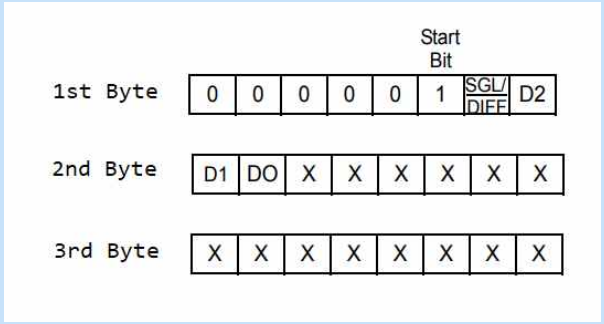


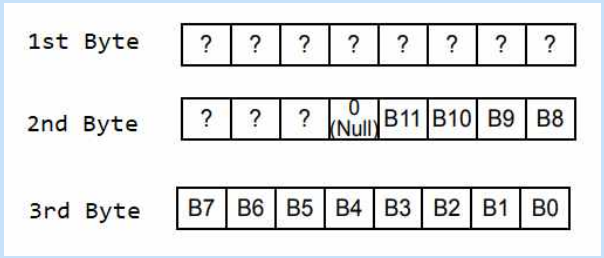


```c
#include <stdio.h>
#include <wiringPi.h>

#define SPI_CH 0
#define ADC_CH 2
#define ADC_CS 29
#define SPI_SPEED 500000     // SPI communication rate

int main(void){
        int value=0, i;
        unsigned char buf[3];

        if(wiringPiSetup() == -1) return 1;

        if(wiringPiSPISetup() == -1) return -1;

        pinMode(ADC_CS,OUTPUT);

        for(i=0; i<20; i++){
// 0000 0 start single D2  buf[0] = 0x06 | ((ADC_CH & 0x04)>>2);
// D1 D0 00 0000           buf[1] = ((ADC_CH & 0x03)<<6);
// 0000 0000               buf[2] = 0x00;

                digitalWrite(ADC_CS,0);

                wiringPiSPIDataRW(SPI_CH,buf,3);

                buf[1]=0x0F & buf[1];    // masking lower nibble

// 12 bits by concatenation value = (buf[1]<<8) | buf[2];

                digitalWrite(ADC_CS,1);

                printf("%d\n",value);

                delay(100);
        }
}
```

wiringPiSPIDataRW performs a simultaneous write-read transaction over the selected SPI bus.

Data that was in your buffer is overwritten by data returned from the SPI bus.





```
pi@raspberrypi:~ $ gcc -o SMART_SOUND SMART_SOUND.c -lwiringPi
pi@raspberrypi:~ $ sudo ./SMART_SOUND
51
31
29
366
78
324
39
492
50
463
45
721
2622
499
1666
251
933
121
562
59
```

[Figure 3-19] Compiling and Running *SMART_SOUND* File

Access and Control of IoT Devices

- Light Sensors are photoelectric devices that convert light energy whether visible or infra-red light into an electrical signal

  - Auto screen brightness adjustments

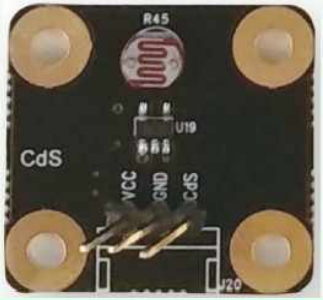
  - Automatically turn on light systems if getting dark

Access and Control of IoT Devices

A CDS photocell or Light Dependant Resistor is a resistor where the resistance changes based on the amount of light. As the amount of light increases the resistance of the sensor decreases and vice versa.

CDS = **Cadmium Sulfide**

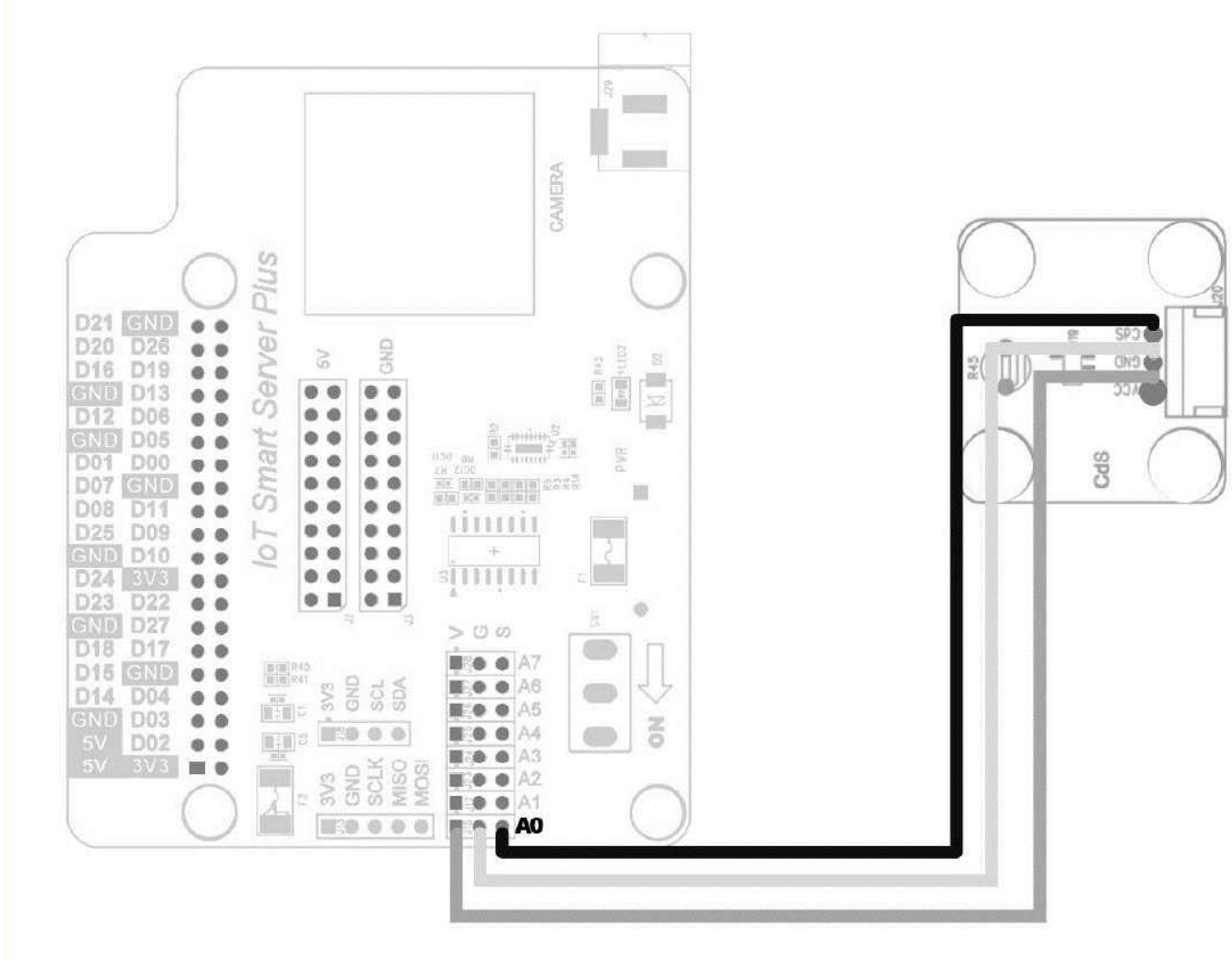
<Table 3-17> Specifications of Light Sensor Module

Shape	Category	Description
	Sensor	Light(CDS)
	Interface	1pin Analog OUTPUT
	Operating Voltage	5V

<Table 3-18> Pin Connection Information of SPI ADC and Light Sensor Module

ADC Port	Light Sensor Pin No.
ADC0	CdS

Connect module without applying power to RPi



```c
#include <stdio.h>
#include <wiringPi.h>

 #define SPI_CH 0
 #define ADC_CH 0
 #define ADC_CS 29
 #define SPI_SPEED 500000

int main(void){
        int value=0, i;
        unsigned char buf[3];

        if(wiringPiSetup() == -1) return 1;
        if(wiringPiSPISetup() == -1) return -1;

        pinMode(ADC_CS,OUTPUT);

        for(i=0; i<20; i++){
                buf[0] = 0x06 | ((ADC_CH & 0x04)>>2);
                buf[1] = ((ADC_CH & 0x03)<<6);
                buf[2] = 0x00;

                digitalWrite(ADC_CS,0);

                wiringPiSPIDataRW(SPI_CH,buf,3);

                buf[1]=0x0F & buf[1];

                value=(buf[1] << 8) | buf[2];

                digitalWrite(ADC_CS,1);

                printf("%d\n", value);
                delay(100);
        }
}

```

If you block the light near the sensor, the output of the sensor becomes low.

If the light is bright, the output is high.

```
pi@raspberrypi:~ $ gcc -o SMART_LIGHT SMART_LIGHT.c -lwiringPi
pi@raspberrypi:~ $ sudo ./SMART_LIGHT
1359
1353
1357
1352
1315
1252
1139
963
208
183
98
224
183
177
187
183
216
143
896
1222
```

Access and Control of IoT Devices

- Gas Sensor detect and identify different types of gasses
  - Gas sensors are employed in factories and manufacturing facilities to identify gas leaks, and to detect smoke and carbon monoxide in homes
  - MQ-6 gas sensor can detect kinds of flammable gases, especially has high sensitivity to LPG

<Table 3-46> Specifications of Gas Sensor Module

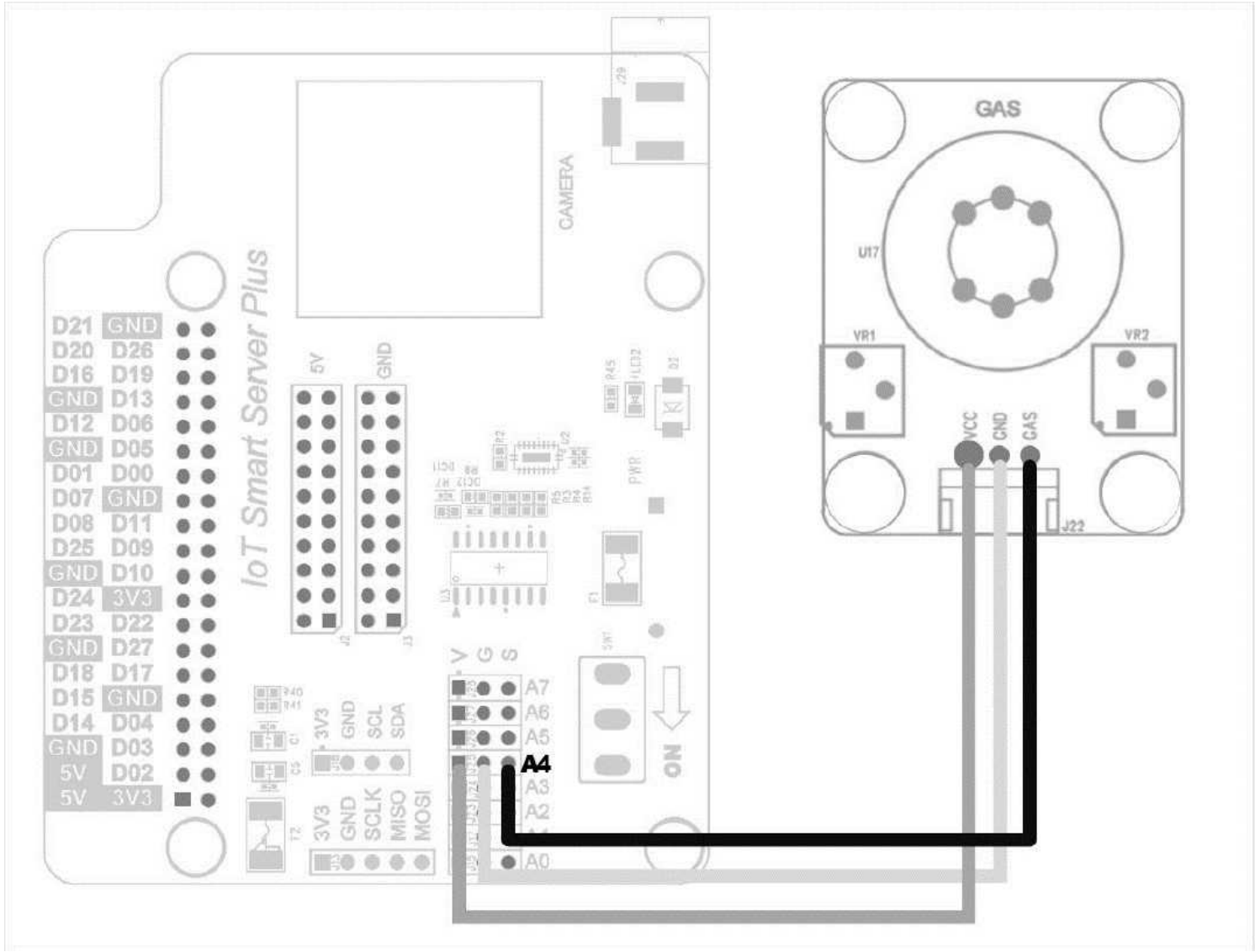
Shape	Category	Description
	Sensor	Gas Sensor
	I/O Interface	1pin Analog OUTPUT
	Operating Voltage	5V

https://components101.com/sensors/mq-6-gas-sensor-pinout-equivalent-datasheet

〈Table 3-47〉 Pin Connection Information for SPI ADC and Gas Sensor Module

ADC Port	Gas Sensor Module Pin No.
ADC4	GAS

Connect module without applying power to RPi



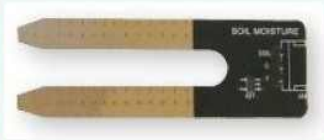
```c
#include <stdio.h>
#include <wiringPi.h>

#define SPI_CH 0
#define ADC_CH 4
#define ADC_CS 29
#define SPI_SPEED 500000

int main(void){
        int value=0, i;
        unsigned char buf[3];

        if(wiringPiSetup() == -1) return 1;

        if(wiringPiSPISetup() == -1) return -1;

        pinMode(ADC_CS,OUTPUT);

        for(i=0; i<20; i++){
                buf[0] = 0x06 | ((ADC_CH & 0x04)>>2);
                buf[1] = ((ADC_CH & 0x03)<<6);
                buf[2] = 0x00;

                digitalWrite(ADC_CS,0);

                wiringPiSPIDataRW(SPI_CH,buf,3);

                buf[1]=0x0F & buf[1];

                value=(buf[1] << 8) | buf[2];

                digitalWrite(ADC_CS,1);

                printf("%d\n", value);
                delay(100);
        }
}
```

```
pi@raspberrypi:~ $ gcc -o SMART_GAS SMART_GAS.c -lwiringPi
pi@raspberrypi:~ $ sudo ./SMART_GAS
722
723
723
723
722
722
721
722
722
720
720
718
719
720
719
718
718
714
720
```

Access and Control of IoT Devices

- Soil Moisture Sensor is used for measuring the moisture in soil and similar materials
  - Smart farming systems

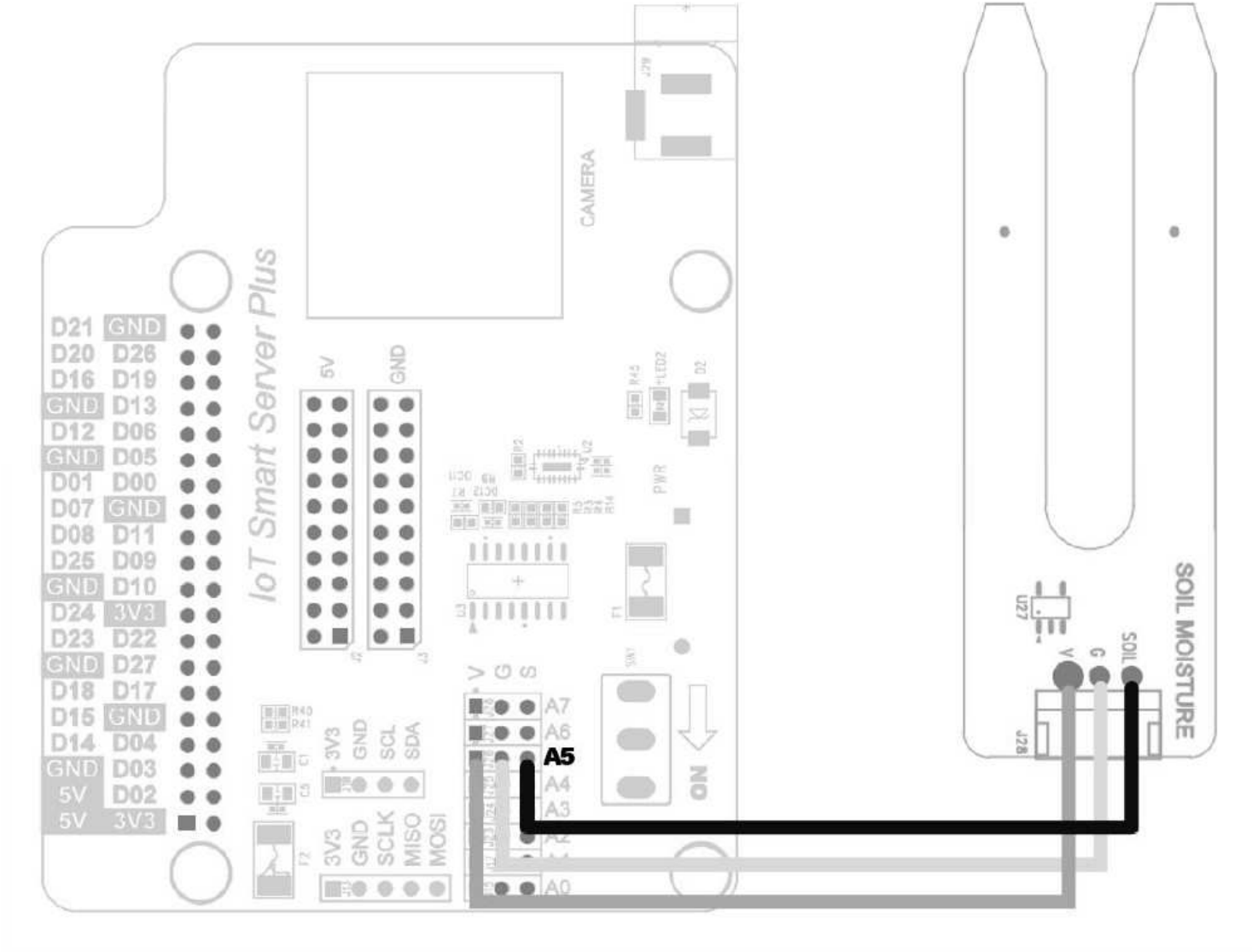
<Table 3-52> Specfications of Soil Moisture Sensor Module

Shape	Category	Description
	Sensor	Soil Moisture Sensor
	I/O Interface	1pin Analog OUTPUT
	Operating Voltage	3.3V~5V

<Table 3-53> Pin Connection Information for SPI ADC and Soil Moisture Sensor

ADC Port	Soil Moisture Sensor Pin No.
ADC5	SOIL

Connect module without applying power to RPi



```c
#include <stdio.h>
#include <wiringPi.h>

#define SPI_CH 0
#define ADC_CH 5
#define ADC_CS 29
#define SPI_SPEED 500000

int main(void){
        int value=0, i;
        unsigned char buf[3];

        if(wiringPiSetup() == -1) return 1;

        if(wiringPiSPISetup() == -1) return -1;

        pinMode(ADC_CS,OUTPUT);


        for(i=0; i<20; i++){
                buf[0] = 0x06 | ((ADC_CH & 0x04)>>2);
                buf[1] = ((ADC_CH & 0x03)<<6);
                buf[2] = 0x00;

                digitalWrite(ADC_CS,0);

                wiringPiSPIDataRW(SPI_CH,buf,3);

                buf[1]=0x0F & buf[1];

                value=(buf[1] << 8) | buf[2];

                digitalWrite(ADC_CS,1);

                printf("%d\n", value);
               delay(100);
        }
}
```

```
pi@raspberrypi:~ $ gcc -o SMART_SM SMART_SM.c -lwiringPi
pi@raspberrypi:~ $ sudo ./SMART_SM
60
60
61
61
60
61
165
434
1239
697
2263
69
2240
69
68
68
68
70
68
68
```

Any Questions!