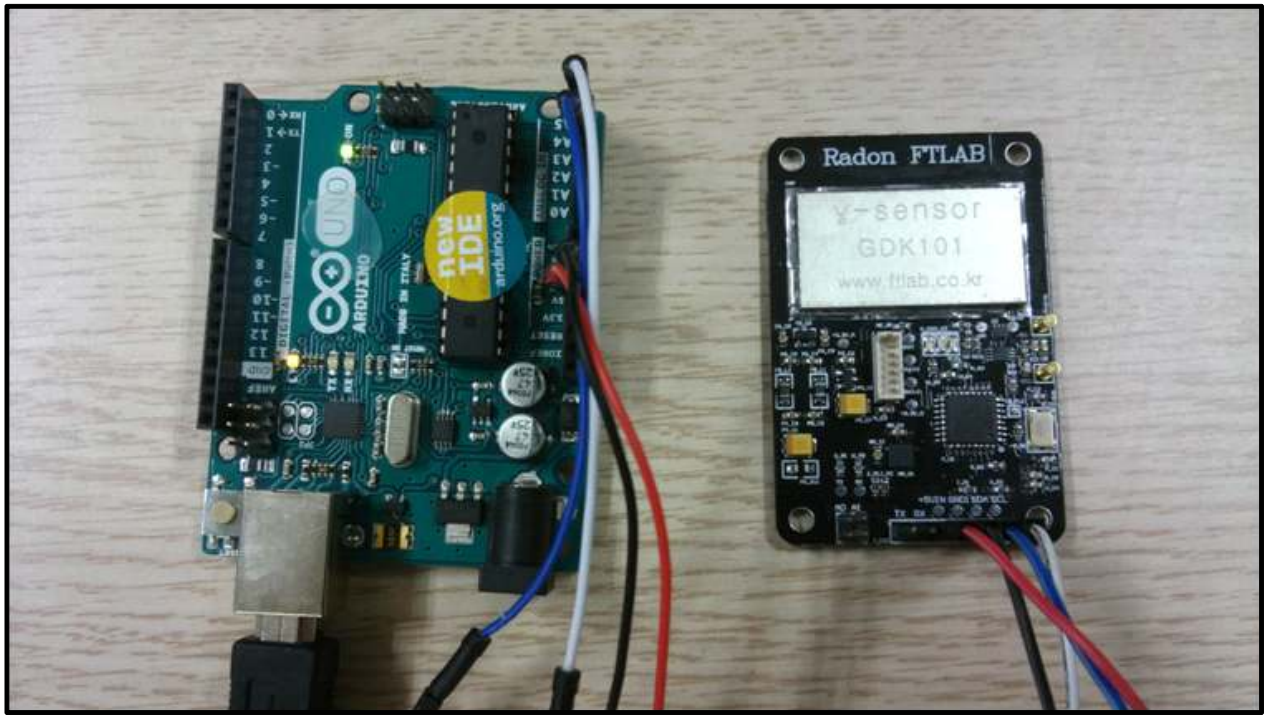
Application Note: Interfacing with Arduino over I²C

The Arduino makes an ideal platform for prototyping and data collection with the Gamma sensors.

Electrical Connections

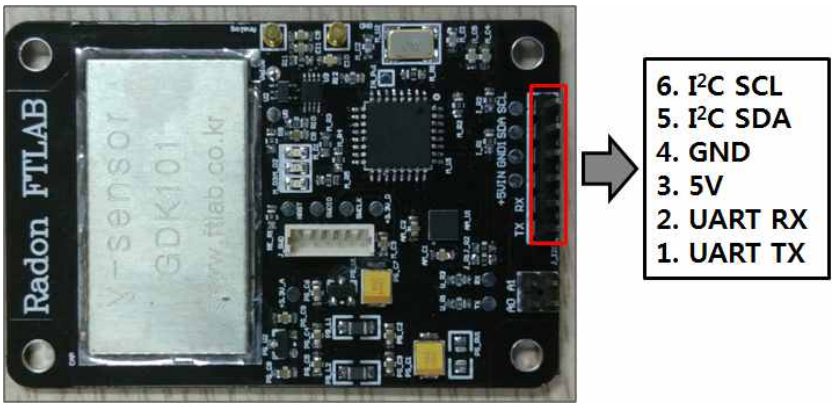


Interfacing with the sensor electrically is easy. A direct electrical connection is possible using the Arduino's hardware I²C pins as follows:

* Arduino analog input A5 - I²C SCL
* Arduino analog input A4 - I²C SDA

Both the Arduino and Gamma sensor have built-in pull-up resistors.

The sensor will be wired using the I²C terminal in the following drawing:



Gamma sensor module GDK101

Software Interface

We will use the built in Wire library to interface with the Gamma sensor. Import it into the project and initialize it in the setup() routine:

```
void setup() {
  //Arduino Initialize
  Serial.begin(9600);
  Wire.begin();
  Serial.println("Gamma Sensor Sensing Start");

  //Read Firmware Version
  Gamma_Mod_Read(0xB4);
  //Reset before operating the sensor
  Gamma_Mod_Read(0xA0);
  Serial.println("================================================");
}
```

Next we will start polling the sensor using the standard I^2C sequences.

```
void loop() {
  delay(1000);
  Gamma_Mod_Read(0xB0); // Read Status
  Gamma_Mod_Read(0xB1); // Read Measuring Time
  Gamma_Mod_Read(0xB2); // Read Measuring Value (10min avg.)
  Gamma_Mod_Read(0xB3); // Read Measuring Value (1min avg.)
  Serial.println("================================================");
  sec++;
}

void Gamma_Mod_Read(int cmd){
  /* Begin Write Sequence */
  Wire.beginTransmission(addr);
  Wire.write(cmd);
  Wire.endTransmission();
  /* End Write Sequence */
  delay(10);
  /* Begin Read Sequence */
  Wire.requestFrom(addr, 2);
  byte i = 0;
  while(Wire.available())
  {
    buffer[i] = Wire.read();
    i++;
  }
  /* End Read Sequence */

  /* View Results */
  Print_Result(cmd);
}
```

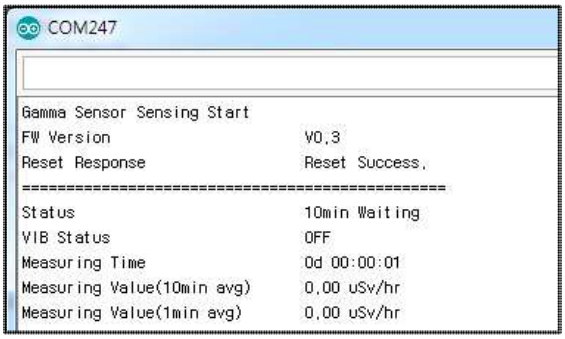
Appendix A: Sample Code

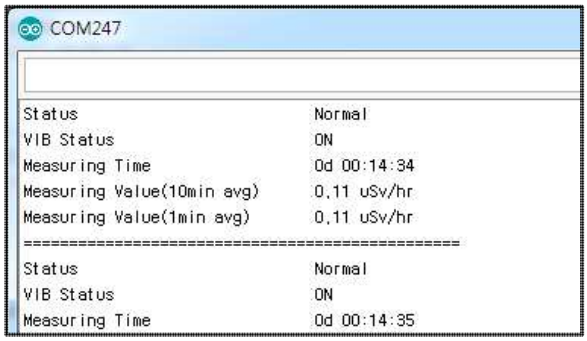
```
// Gamma Sensor's Example Interface

#include <Wire.h>
/*
 *    We will be using the I2C hardware interface on the Arduino in
 *    combination with the built-in Wire library to interface.
 *    Arduino analog input 5 - I2C SCL
 *    Arduino analog input 4 - I2C SDA
 *
 *    Command List
 *    0xA0   ::   Reset
 *    0xB0   ::   Read Status
 *    0xB1   ::   Read Measuring Time
 *    0xB2   ::   Read Measuring Value (10min avg / 1min update)
 *    0xB3   ::   Read Measuring Value (1min avg / 1min update)
 *    0xB4   ::   Read Firmware Version
 *
 *    Address Assignment
 *    Default Address        :: 0x18
 *    A0 Open, A1 Short   :: 0x19
 *    A0 Short, A1 Open   :: 0x1A
 *    A0 Open, A1 Open   :: 0x1B
 */
int addr = 0x18;
int day,hour,min,sec = 0;
byte buffer[2] = {0,0};
int status = 0;

void setup() {
  //Arduino Initialize
  Serial.begin(9600);
  Wire.begin();
  Serial.println("Gamma Sensor Sensing Start");

  //Read Firmware version
  Gamma_Mod_Read(0xB4);
  //Reset before operating the sensor
  Gamma_Mod_Read(0xA0);
  Serial.println("================================================");
}

void loop() {
  delay(1000);
  //Read Statue, Measuring Time, Measuring Value
  Gamma_Mod_Read_Value();
  Serial.println("================================================");
}
```

```
void Gamma_Mod_Read_Value(){
  Gamma_Mod_Read(0xB0); // Read Status
  Gamma_Mod_Read(0xB1); // Read Measuring Time
  Gamma_Mod_Read(0xB2); // Read Measuring Value (10min avg / 1min update)
  Gamma_Mod_Read(0xB3); // Read Measuring Value (1min avg / 1min update)
}

void Gamma_Mod_Read(int cmd){
  /* Begin Write Sequence */
  Wire.beginTransmission(addr);
  Wire.write(cmd);
  Wire.endTransmission();
  /* End Write Sequence */
  delay(10);
  /* Begin Read Sequence */
  Wire.requestFrom(addr, 2);
  byte i = 0;
  while(Wire.available())
  {
    buffer[i] = Wire.read();
    i++;
  }
  /* End Read Sequence */

  /* View Results */
  Print_Result(cmd);
}

/*
 *  Calculation Measuring Time
 *  Format :: 0d 00:00:00 ( (day)d (hour):(min):(sec) )
 */
void Cal_Measuring_Time(){
  if(sec == 60)   { sec = 0;  min++;  }
  if(min == 60)   { min = 0;  hour++; }
  if(hour == 24)  { hour = 0; day++;  }

  Serial.print("Measuring Time\t\t\t");
  Serial.print(day); Serial.print("d ");
  if(hour < 10) Serial.print("0");
  Serial.print(hour); Serial.print(":");
  if(min < 10) Serial.print("0");
  Serial.print(min); Serial.print(":");
  if(sec < 10) Serial.print("0");
  Serial.println(sec);
}
```

```
void Print_Result(int cmd){
  float value = 0.0f;
  switch(cmd){
    case 0xA0:
      Serial.print("Reset Response\t\t\t");
      if(buffer[0]== 1) Serial.println("Reset Success.");
      else              Serial.println("Reset Fail(Status - Ready).");
      break;

    case 0xB0:
      Serial.print("Status\t\t\t\t");
      switch(buffer[0]){
        case 0: Serial.println("Ready"); break;
        case 1: Serial.println("10min Waiting"); break;
        case 2: Serial.println("Normal"); break;
      }
      status = buffer[0];
      Serial.print("VIB Status\t\t\t");
      switch(buffer[1]){
        case 0: Serial.println("OFF"); break;
        case 1: Serial.println("ON"); break;
      }
      break;

    case 0xB1:
      if(status > 0){
        sec++;
        Cal_Measuring_Time();
      }
      break;

    case 0xB2:
      Serial.print("Measuring Value(10min avg)\t");
      value = buffer[0] + (float)buffer[1]/100;
      Serial.print(value); Serial.println(" uSv/hr");
      break;

    case 0xB3:
      Serial.print("Measuring Value(1min avg)\t");
      value = buffer[0] + (float)buffer[1]/100;
      Serial.print(value); Serial.println(" uSv/hr");
      break;

    case 0xB4:
      Serial.print("FW Version\t\t\t");
      Serial.print("V"); Serial.print(buffer[0]);
      Serial.print("."); Serial.println(buffer[1]);
      break;
  }
}
```

Result





Command List

START	5-bit Address	2-bit User address	Write bit	Command	START	5-bit Address	2-bit User address	Read bit	Read data 1	Read data 2	STOP
	default addr. + user addr.		0	CMD		default addr. + user addr.		1			
	1 byte			1 byte		1 byte			1 byte	1 byte	

* Command

CMD	Description	Read Data 1	Read Data 2
0xA0	Reset	0 - Fail 1 - Pass	Not used
0xB0	Read the status of measurement and vibration	0 - Power On ~ 10sec 1 - 10sec to 10min 2 - After 10 min	0 - Not detect vibrations 1 - Detect vibrations
0xB1	Read Measuring Time	Minutes of time	Seconds of time
0xB2	Read Measured Value (10min avg. 1min update)	Integer of value	Decimal of value
0xB3	Read Measured Value (1min avg. 1min update)	Integer of value	Decimal of value
0xB4	Read Firmware Version	Main of version	Sub of version

* Address Setting

- Address assignment

No.	A0	A1	Address
1	Short	Short	**0x18**
2	Open	Short	**0x19**
3	Short	Open	**0x1A**
4	Open	Open	**0x1B**

- Since you can only have one device with a given address on an I^2C bus, there must be a way to adjust the address if you want to put more than one Gamma sensor on a shared I^2C bus.
- They are read on power up, so de-power and re-power to reset the address.
- I^2C Maximum Speed is 100 kHz.