Application Note: Interfacing with Arduino over UART

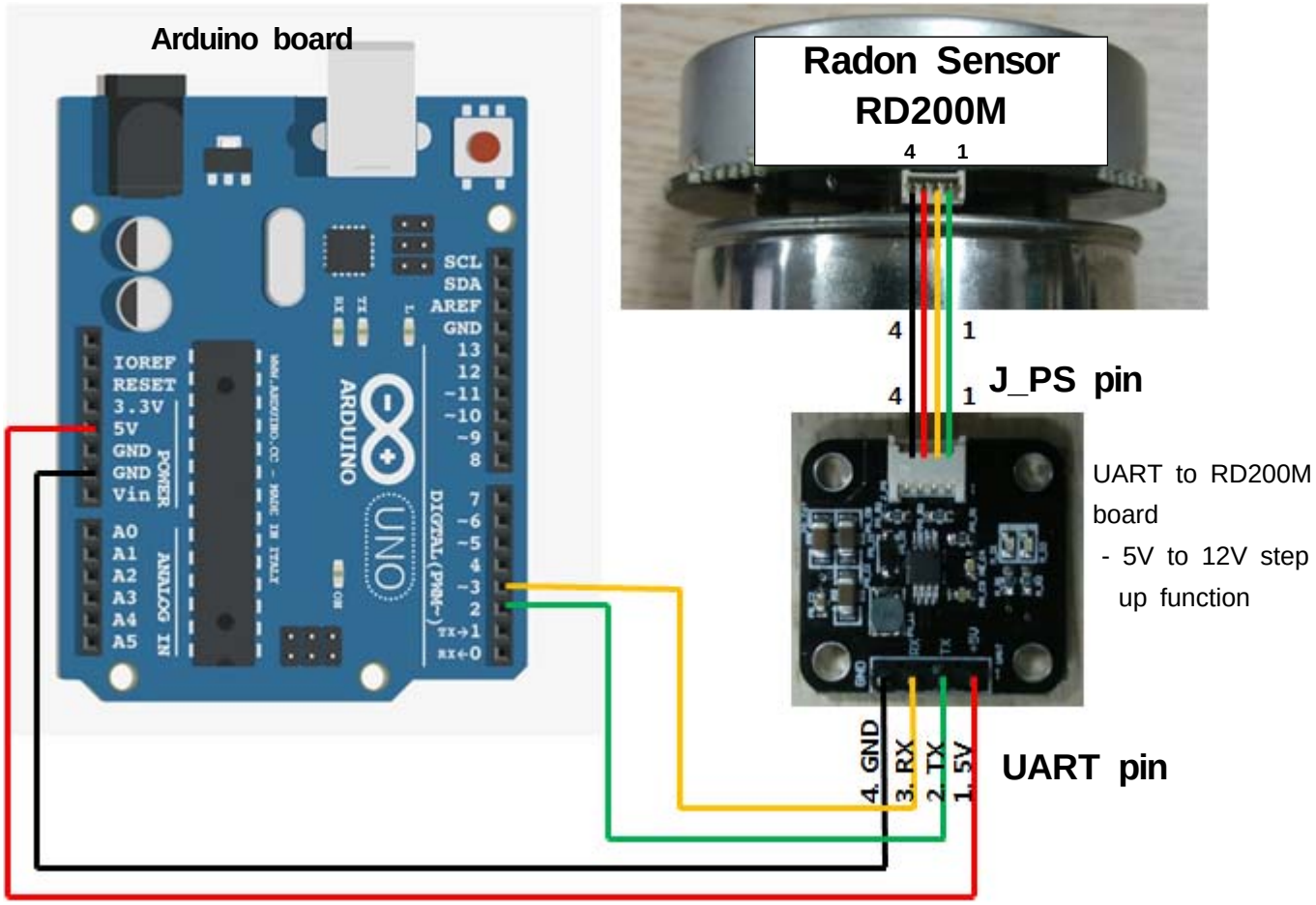
The Arduino makes an ideal platform for prototyping and data collection with the RD200M.

Wiring

The first step is to wire up the RD200M to your Arduino. This example shows the Arduino UNO with a RD200M using a interface board. The interface board, UART to RD200M has a function of 5V to 12V step up

* Make the following Connections:

Wire	Arduino	UART to RD200M Board		RD200M
Red	5V	UART pin #1	J_PS pin #3	pin #3, 12V
Black	GND	UART pin #4	J_PS pin #4	pin #4, GND
Green	Digital pin #2	UART pin #2	J_PS pin #1	pin #1, TX
Yellow	Digital pin #3	UART pin #3	J_PS pin #2	pin #2, RX



Writing the Code

The next step is to write the code to drive the device. We will show a quick and simple sketch that will report the radon value, and then we will show a sketch that does the same job.

We will use SoftwareSerial library, Timer library and mthread library to interface with the RD200 Module sensor. Import it into the project and initialize it in the setup() routine:

```
/* Thread class for receive UART */
class FooThread : public Thread
{
  public: FooThread(int id);
  protected: bool loop();
  private:  int id;
};

FooThread::FooThread(int id){
  this->id = id;
}

/* Pin map :: Digital 2 - RX (to RD200M TX), Digital 3 - TX (to RD200M RX) */
SoftwareSerial mySerial(2, 3);
Timer ts1, ts2;
void setup() {
  Serial.begin(19200);                    // PC - Arduino
  mySerial.begin(19200);                  // Arduino - RD200M
  ts1.every(1000, RequestData);           // Timer 1 : Request to RD200M
  ts2.every(1000, CalMeasuringTime);      // Timer 2 : Measuring Time

  main_thread_list->add_thread(new FooThread(1));     // Add Thread 1 for receive data
  RD200M_INIT();
}
```

We will start the operation of the RD200M using the threads.

```
bool FooThread::loop()
{
  // Thread 1 - for Receive Data
  if(id == 1){
      RecUartData();
  }
  // Thread 2 - for Timers
  else{
    if(status > 0) ts2.update();
    ts1.update();
  }
  return true;
}
```

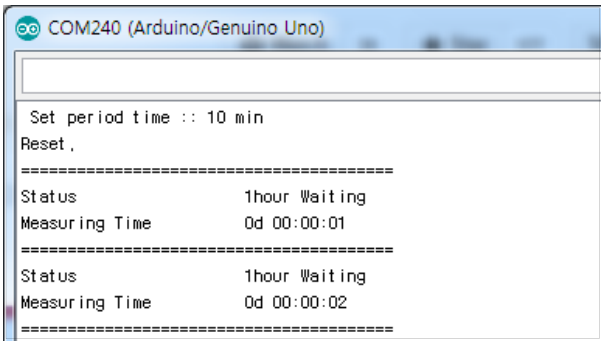
Appendix A: Sample Code

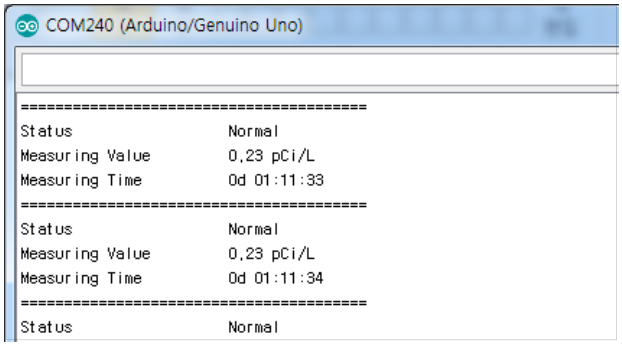
```
#include <SoftwareSerial.h>
#include <Timer.h>        // https://github.com/JChristensen/Timer
#include <mthread.h>      // https://github.com/jlamothe/mthread

/* Thread class for receive UART */
class FooThread : public Thread
{
  public: FooThread(int id);
  protected: bool loop();
  private: int id;
};

FooThread::FooThread(int id){
  this->id = id;
}

/* Command List */
byte cmdRD200M_RESULT_QUERY = 0x01;       // Request all data
byte cmdRD200M_RESET = 0xA0;              // RD200M reset
byte cmdRD200M_SEND_PERIOD_SET = 0xA1;    // Set data transfer period
byte cmdRD200M_RESULT_RETURN = 0x10;      // Read all data (receive only)

/*
    Pin map
    Digital 2 - RX (to RD200M TX)
    Digital 3 - TX (to RD200M RX)
*/
SoftwareSerial mySerial(2, 3);

Timer ts1, ts2;
int status = 0;                           // RD200M status
int day, hour, min, sec, secR = 0;        // Measuring time
byte cmd = 0;                             // UART Command
int period_time = 10;                     // parameters for setting period time, unit:minute
float value = 0.0f;                       // Radon value
byte rec_data[8];                         // Array for received data

/* Calculate checksum */
bool checkSum(byte data[], int rec_size) {
  int sum = 0;
  for (int i = 1; i < rec_size - 1; i++) {
    sum += data[i];
    sum %= 256;
  }
  if (255 - sum == data[rec_size - 1]) return true;
  else return false;
}
```

```
void setup() {
  Serial.begin(19200);                    // PC - Arduino
  mySerial.begin(19200);                  // Arduino - RD200M
  ts1.every(1000, RequestData);           // Timer 1 : Request to RD200M
  ts2.every(1000, CalMeasuringTime);      // Timer 2 : Measuring Time

  main_thread_list->add_thread(new FooThread(1));    // Add Thread 1 for receive data
  RD200M_INIT();
}

bool FooThread::loop()
{
  // Thread 1 - for Receive Data
  if(id == 1){ RecUartData(); }
  // Thread 2 - for Timers
  else{
    if(status > 0) ts2.update();
    ts1.update();
  }
  return true;
}

/* RD200M Initialize  */
void RD200M_INIT() {
  cmd = cmdRD200M_SEND_PERIOD_SET;
  byte send_data[5] = {0x02, cmd, 0x01, period_time, 0xFF - (cmd + 0x01 + period_time)};
  mySerial.write(send_data, 5);
  Serial.print("Set period time :: "); Serial.print(period_time); Serial.println(" min");
  delay(1000);
  cmd = cmdRD200M_RESET;
  RequestData();
  Serial.println("Reset.");
  delay(1000);
  cmd = cmdRD200M_RESULT_QUERY;
  main_thread_list->add_thread(new FooThread(2));    // Add Thread 2 for Timers
}

/* Request to RD200M */
void RequestData() {
  byte send_data[4] = {0x02, cmd, 0x00, 0xFF - cmd}; // STX | CMD | SIZE | CHECKSUM
  mySerial.write(send_data, 4);
}

/* Receive from RD200M */
void RecUartData(){
  int rec_size = mySerial.available();
  if (rec_size == 8) {
    for (int i = 0; i < rec_size; i++) {
      rec_data[i] = (byte)mySerial.read();
    }
```

```
    if (checkSum(rec_data, rec_size) == false) return;
    Serial.println("========================================");

    // View Status
    Serial.print("Status\t\t\t");
    status = rec_data[3];
    switch (status) {
      case 0x00: Serial.println("Ready - Wait 200 sec"); break;      // Power On ~ 200sec
      case 0x01: Serial.println("1hour Waiting"); break;             // 200sec to 1hour
      case 0x10: Serial.println("1hour Waiting(Warning)"); break;    // Measuring time is within 30min and radon count
                                                                     // over is 10. (warning status)
      case 0x02: Serial.println("Normal"); break;                   // After 1hour
      case 0xE0: Serial.println("VIBRATION!"); break;               // Detect vibrations
    }

    // View Measuring Value
    if(min%10 == 0 && sec == 0){  // Values updated every 10 minutes
      value = rec_data[5] + (float)rec_data[6]/100;
    }
    if(min / 30 == 1 || status > 1){
      Serial.print("Measuring Value\t\t");
      Serial.print(value); Serial.println(" pCi/L");
    }
  }
}

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

  Serial.print("Measuring Time\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);
}
```

Result





Command Sequence

* Read all data

Request :

Description	STX 1-byte	Command 1-byte	Data size 1-byte	Checksum 1-byte
Example	0x02	0x01	0x00	0xFE

a. 0x02 : STX
b. 0x01 : Request all data command
c. 0x00 : Data size is 0 bytes
d. 0xFE : Checksum, 0xFE = 0xFF - (0x01+0x00)

Response :

Description	STX 1-byte	Command 1-byte	Data size 1-byte	Data 1 1-byte	Data 2 1-byte	Data 3 1-byte	Data 4 1-byte	Checksum 1-byte
Example	0x02	0x10	0x04	0x01	0x1E	0x02	0x3A	0x90

a. 0x02 : STX
b. 0x10 : Read all data command
c. 0x04 : Data size is 4 bytes
d. 0x01(Data1) : RD200M status 1, it is '200sec to 1hour' after start.
e. 0x1E(Data2) : Minutes of measured time, $1E_{(16)} = 30_{(10)}$
f. 0x02(Data3) : Integer of measured value, $2_{(16)} = 2_{(10)}$
g. 0x3A(Data4) : Decimal of measured value, $3A_{(16)} = 58_{(10)}$
h. 0x90 : Checksum, 0x90 = 0xFF - (0x10+0x04+0x01+0x1E+0x02+0x3A)
--> Measuring time is 30min.
--> Measured value is 2.58 (pCi/L).

* Reset

Request :

Description	STX 1-byte	Command 1-byte	Data size 1-byte	Checksum 1-byte
Example	0x02	0xA0	0x00	0x5F

a. 0x02 : STX

b. 0xA0 : Reset command

c. 0x00 : Data size is 0 bytes

d. 0x5F : Checksum, 0x5F = 0xFF - (0xA0+0x00)

--> No response.

* Set data transfer period

Request :

Description	STX 1-byte	Command 1-byte	Data size 1-byte	Data(minute) 1-byte	Checksum 1-byte
Example	0x02	0xA1	0x01	0x0A	0x53

a. 0x02 : STX

b. 0xA1 : Set data transfer period command

c. 0x01 : Data size is 1 bytes

d. 0x0A : $0A_{(16)} = 10_{(10)}$, Transfer period is set to 10 minutes.

e. 0x53 : Checksum, 0x53 = 0xFF - (0xA1+0x01+0x0A)

--> No response.