ThingMagic
A DIVISION OF TRIMBLE

C API Compilation on Arduino

Version 1.0

- Verified on: M6e
- Supported OS: Windows and Linux

Government Limited Rights Notice: All documentation and manuals were developed at private expense and no part of it was developed using Government funds.

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose the technical data contained herein are restricted by paragraph (b)(3) of the Rights in Technical Data — Noncommercial Items clause (DFARS 252.227-7013(b)(3)), as amended from time-to-time. Any reproduction of technical data or portions thereof marked with this legend must also reproduce the markings. Any person, other than the U.S. Government, who has been provided access to such data must promptly notify ThingMagic.

ThingMagic, A Division of Trimble
1 Merrill St.
Woburn, MA 01801

09 Revision A
Sept, 2015

ThingMagic
A DIVISION OF TRIMBLE

C API Compilation on Arduino

V1.0

Introduction:

This document describes how to configure, compile C API and run sample application for Arduino Mega 2560.

Download Arduino & API:

Download the Arduino IDE from

- Linux: https://www.arduino.cc/en/Guide/Linux
- Windows: https://www.arduino.cc/en/Guide/Windows

Follow the installation steps as shown in the corresponding web page

- Download Mercury API SDK from http://www.thingmagic.com/images/Downloads/software/mercuryapi-1.27.3.16.zip

***** Note: Download and install Arduino IDE for Linux/Ubuntu from the above mentioned URL instead of Ubuntu software Center.**

API Example & Configuration

Configuring the read.ino

- Open the Arduino IDE and copy the Read.ino sample code.
- If you are using nano module below two changes are required in the 'Read.ino' sample code.

1. Set region to NA3 in initializeReader() as nano doesn't support NA region.

```
region = TMR_REGION_NA3;
```

2. Initialize the simple read plan with specified antenna list in readTags() as nano doesn't have internal antenna detection.

```
uint8_t antennaList[1]= {1};
ret = TMR_RP_init_simple(&plan, antennaCount, &antennaList[0],
TMR_TAG_PROTOCOL_GEN2, 1000);
```

For nano sample code refer Read_nano.ino

- Add required files to Arduino:

Copy the below listed files to one seperate folder from API folder (mercuryapi-1.27.3.16\c\src\api).

1. Hex_bytes.c
2. Osdep.h
3. Osdep_arduino.c
4. Serial_reader.c
5. Serial_reader_imp.h
6. Serial_reader_l3.c
7. Serial_transport_arduino.c
8. Tm_config.h
9. tm_reader.c
10. Tm_reader.h
11. Tm_reader_async.c
12. Tmr_filter.h
13. Tmr_gen2.h
14. Tmr_gpio.h
15. Tmr_ipx.h
16. Tmr_iso180006b.h
17. Tmr_param.c

18. Tmr_params.h
19. Tmr_read_plan.h
20. Tmr_region.h
21. Tmr_serial_reader.h
22. Tmr_serial_transport.h
23. Tmr_status.h
24. tmr_strerror.c
25. Tmr_tag_auth.h
26. Tmr_tag_data.h
27. tmr_tag_lock_action.h
28. Tmr_tagop.h
29. Tmr_tag_protocol.h
30. Tmr_types.h
31. Tmr_utils.h
32. Tmr_utils.c

Mercury API Configuration

A. In the **tm_config.h,** apply the below changes .

- **#define TMR_ENABLE_BACKGROUND_READS -> #undef TMR_ENABLE_BACKGROUND_READS**

undef the below parameter which will disable the background reads which are not in use. By undef this, can disable the <pthread.h>. (this is one of the error we get while Compiling)

- Un comment **#define TMR_ENABLE_SERIAL_READER_ONLY**

- In windows while compiling, we get some warnings. To overcome these undef the following

#define TMR_ENABLE_ERROR_STRINGS-> #undef TMR_ENABLE_ERROR_STRINGS

B. Rename **Serial_transport_arduino.c** to **Serial_transport_arduino.cpp**

C. In **osdep_arduino.c**, comment the below header files & statements.

```
// #include <Time.h>
//#include <WProgram.h>
```

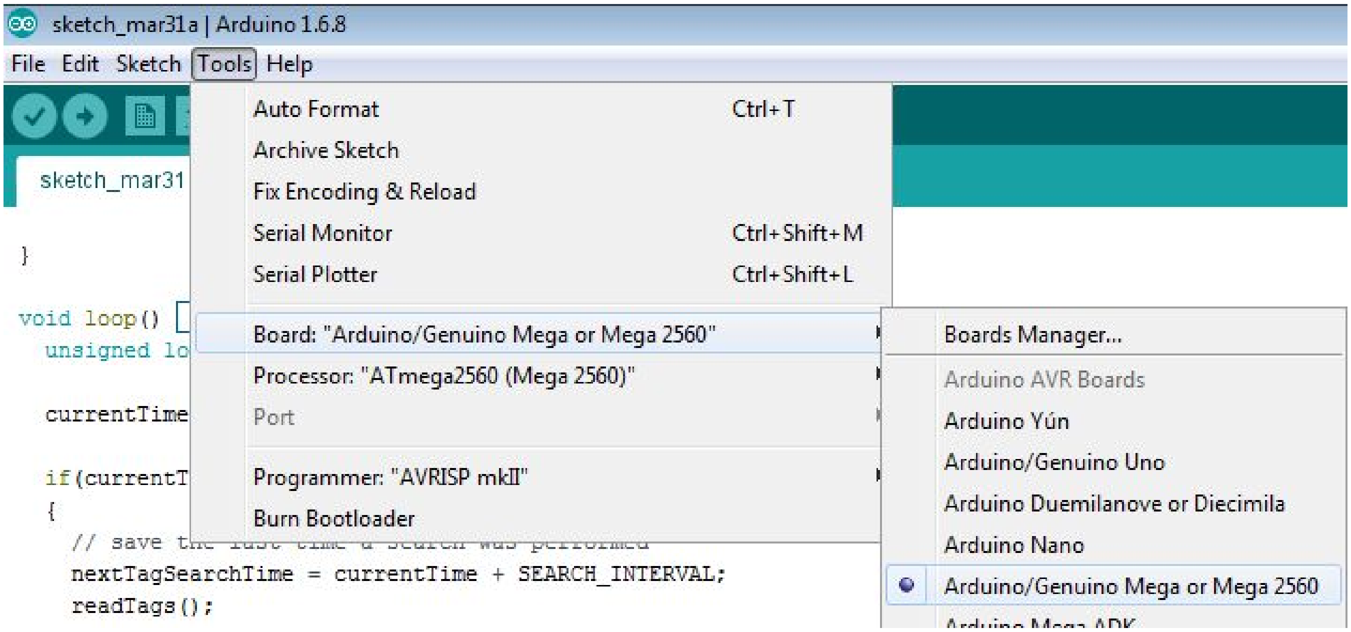
```
/*
TMR_TimeStructure
tmr_gettimestructure()
{
  uint64_t temp;
  time_t now;
  TMR_TimeStructure timestructure;
  static tmElements_t elements;

  temp = tmr_gettime();
  now = temp/1000;

  breakTime(now, elements);  // break time_t into elements
  timestructure.tm_year = (uint32_t)(1990 + elements.Year);
  timestructure.tm_mon = (uint32_t)(1 + elements.Month);
  timestructure.tm_mday = (uint32_t)elements.Day;
  timestructure.tm_hour = (uint32_t)elements.Hour;
  timestructure.tm_min = (uint32_t)elements.Minute;
  timestructure.tm_sec = (uint32_t)elements.Second;
  return timestructure;
}*/
```

***** Note: Wherever you find the # include <WProgram.h> undef/comment it**

● In Arduino IDE, go to Sketch->Include library -> Add zip files and add the folder created in above step.

Compilation Steps

Before Compile the code, select the board ‘Arduino Mega 2560’ as shown below


sketch_mar31a | Arduino 1.6.8
File Edit Sketch Tools Help
sketch_mar31
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
Port
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino Yún
Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino/Genuino Mega or Mega 2560

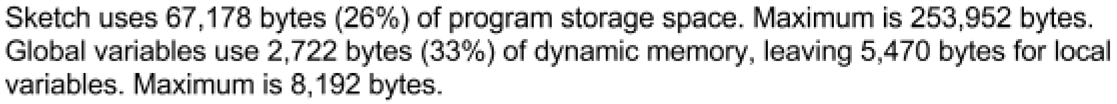
Board: "Arduino/Genuino Mega or Mega 2560"

Processor: "ATmega2560 (Mega 2560)"

Programmer: "AVRISP mkII"

```
}

void loop() 
  unsigned lo

  currentTime

  if(currentT
  {
    // save the last time a search was performed
    nextTagSearchTime = currentTime + SEARCH_INTERVAL;
    readTags();
```

Compile the read.ino file then you will get result as follows



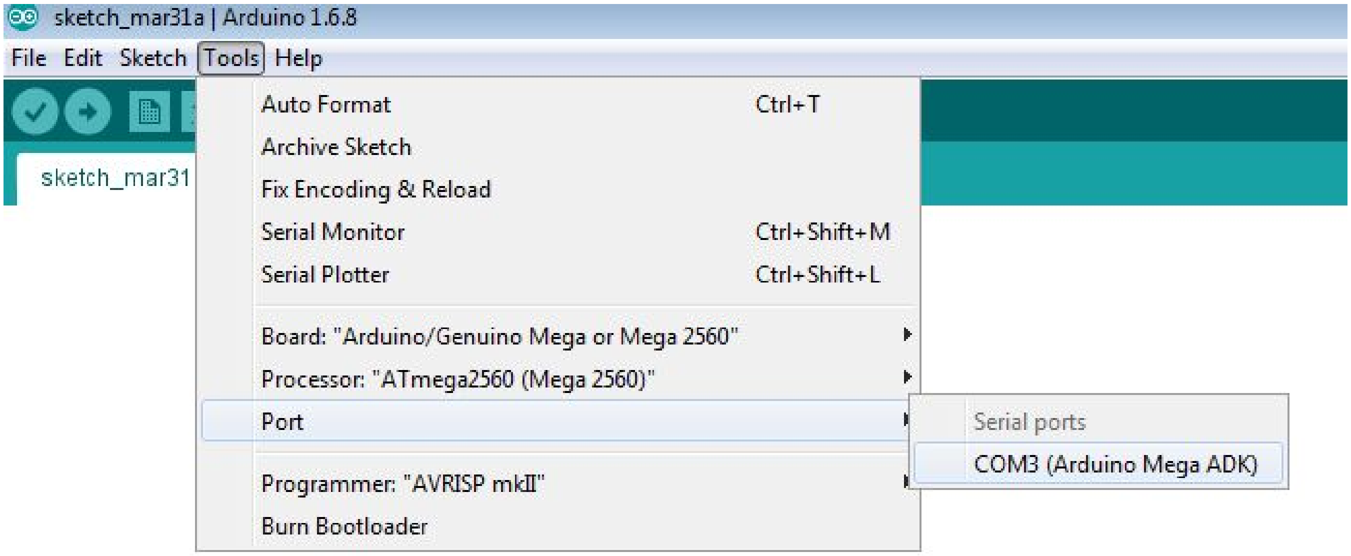
```
Sketch uses 67,178 bytes (26%) of program storage space. Maximum is 253,952 bytes.
Global variables use 2,722 bytes (33%) of dynamic memory, leaving 5,470 bytes for local
variables. Maximum is 8,192 bytes.
```

Upload the Code to Board

After compile the code, we have to upload to the board. While upload please take care of below things.

Windows:

- In Arduino IDE, Go to tools and set PORT shown as below



Board: "Arduino/Genuino Mega or Mega 2560"

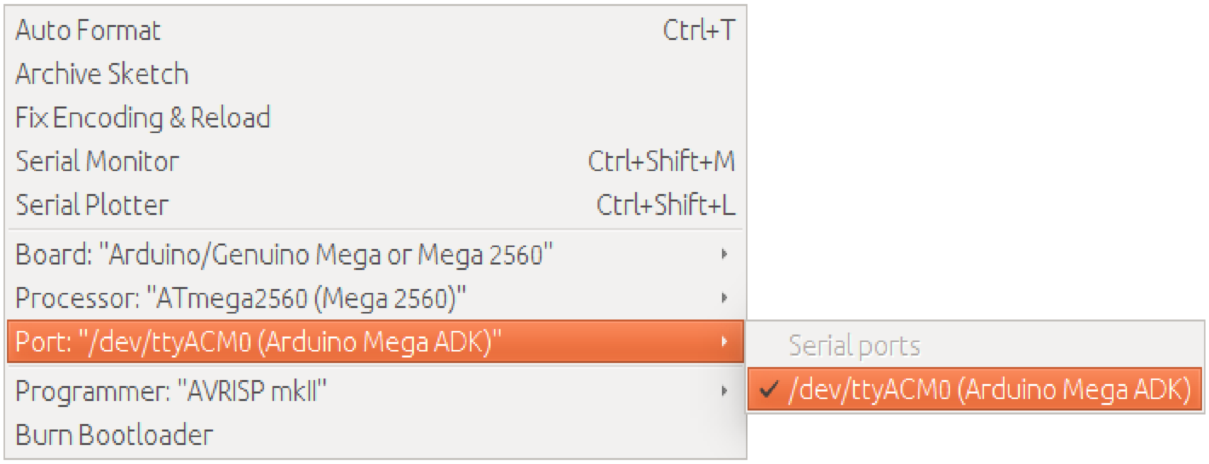
Processor: "ATmega2560 (Mega 2560)"

Port: COM3 (Arduino Mega ADK)

Programmer: "AVRISP mkII"

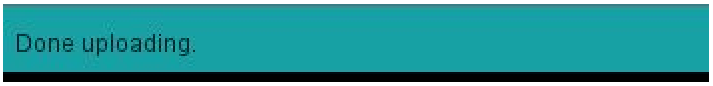
Linux:

- In Arduino IDE, Go to tools and set PORT shown as below.


Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
Board: "Arduino/Genuino Mega or Mega 2560"
Processor: "ATmega2560 (Mega 2560)"
Port: "/dev/ttyACM0 (Arduino Mega ADK)"
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
✓ /dev/ttyACM0 (Arduino Mega ADK)

- Goto Sketch->select the upload option [ctrl+u]

While uploading the code to board, the Receiver & transmitter connected LEDS will blink continuously until upload completes. Once it's completed you can see the below in Arduino IDE.

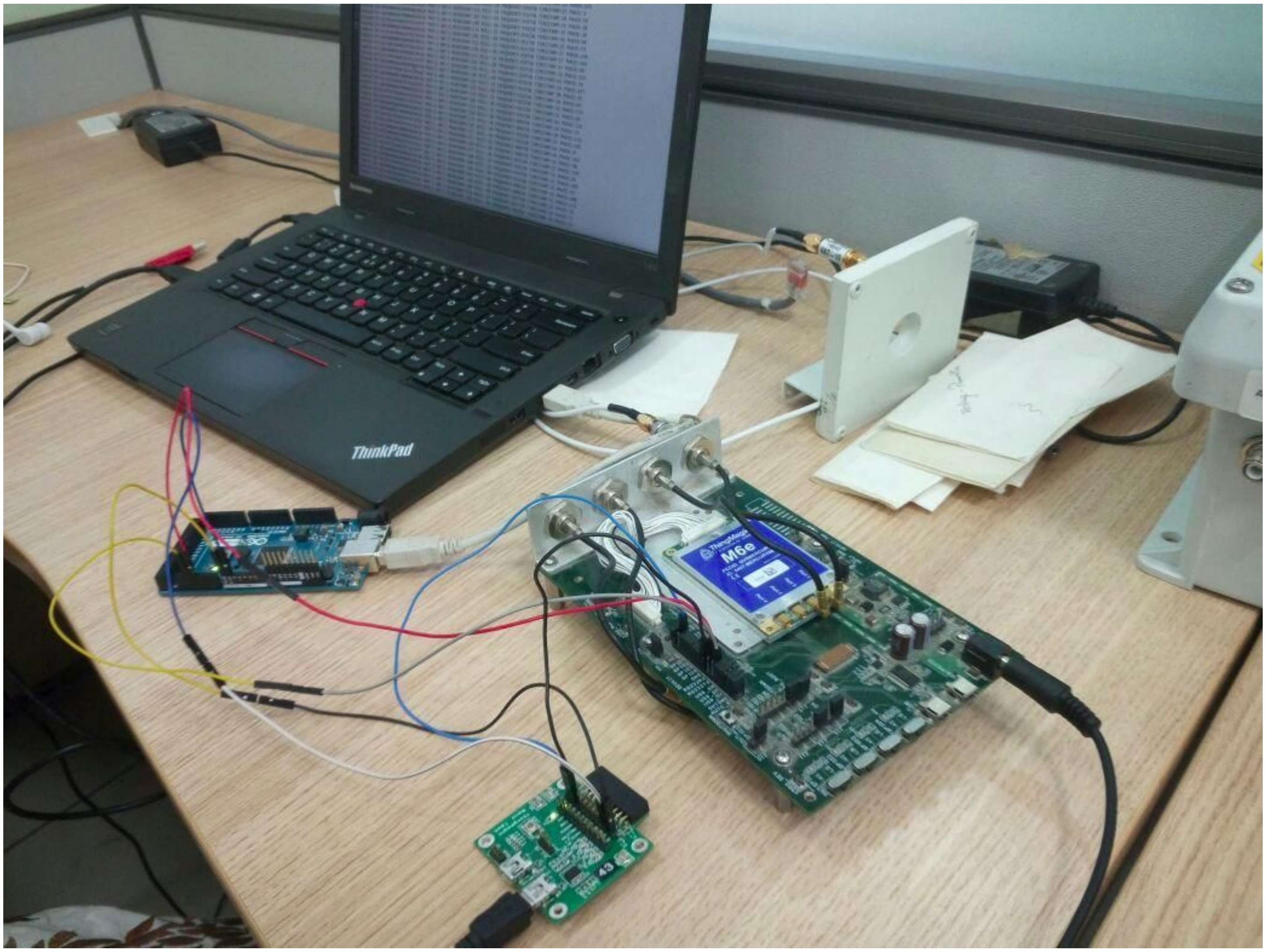


Done uploading.

Hardware Configuration:

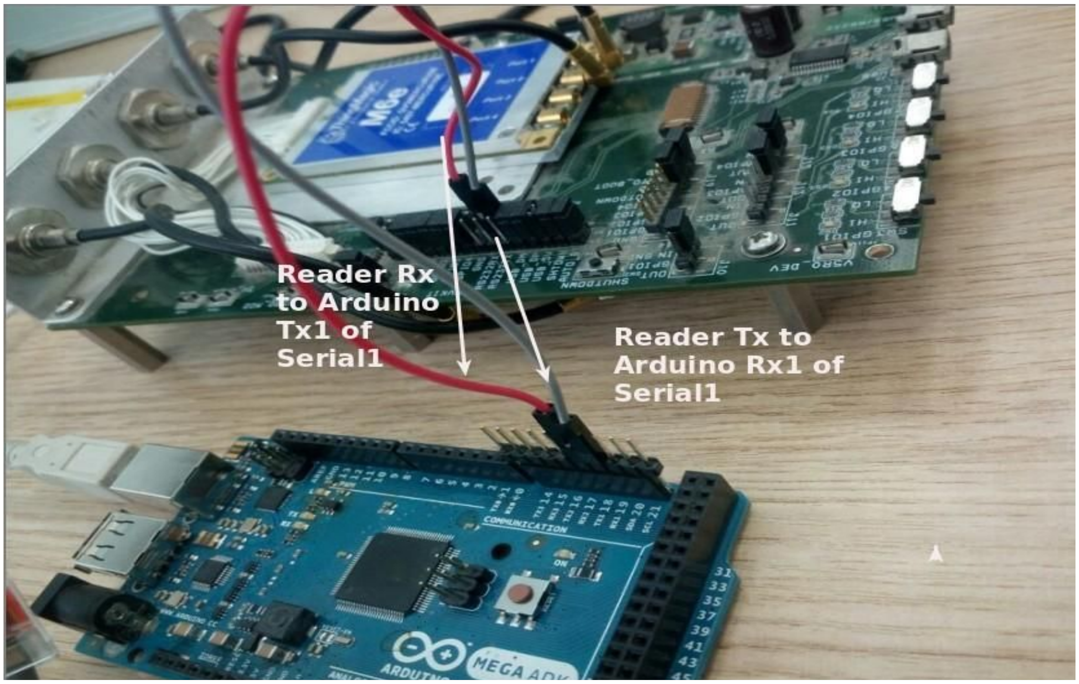
Arduino Hardware Settings:

- Here we are consider Two serial ports Serial 0 & serial 1
- To Serial 0: TM Reader m6e should be connected
- To Serial 1: USB to ftdi is connected



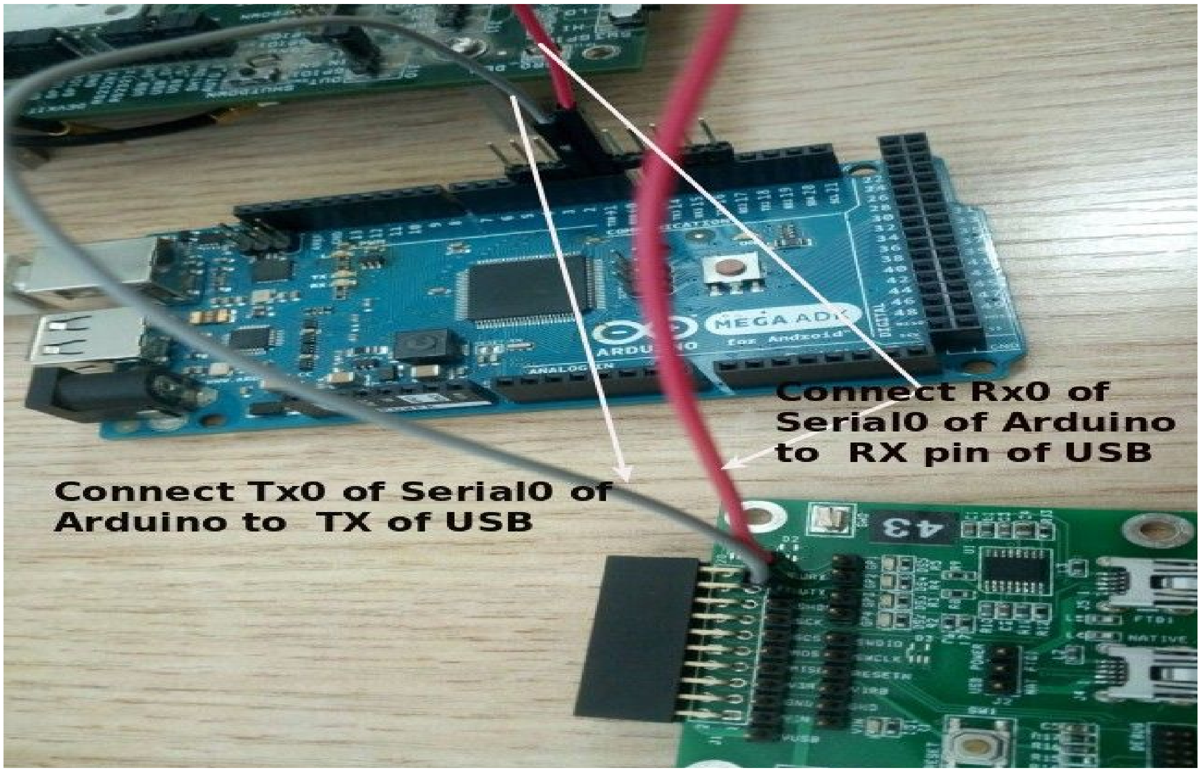
TM Reader Settings:

- Connect the wires as shown in below figures
- Connect wires of Reader to Arduino Serial 1 pins of Transmitter and Receiver
- Antenna can be connected to any one of two antennas


Reader Rx to Arduino Tx1 of Serial1
Reader Tx to Arduino Rx1 of Serial1
MEGA ADK

Hardware Setting of USB to Arduino

- Connect the pins from Arduino to USB as shown in figure

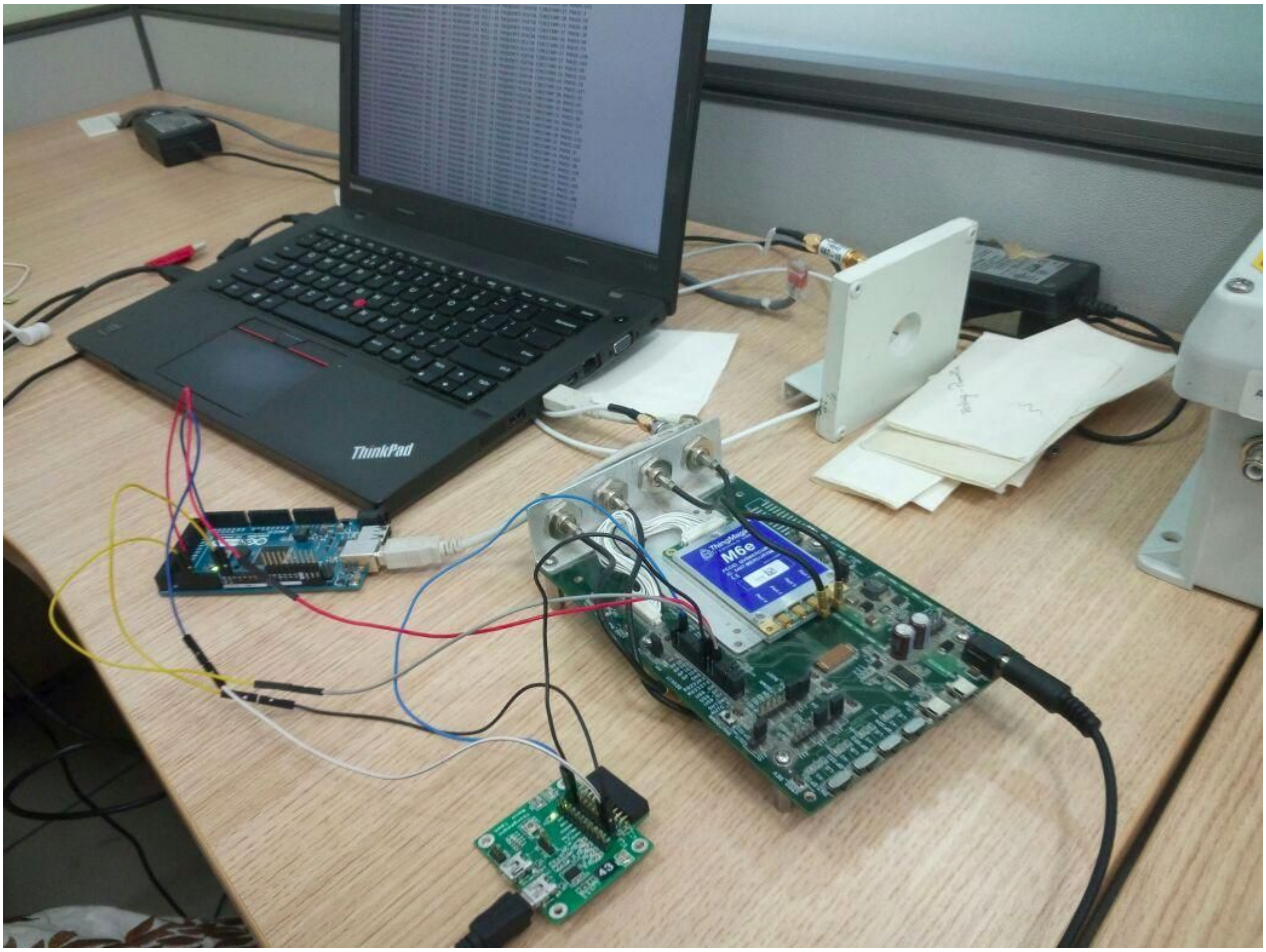


Connect Rx0 of Serial0 of Arduino to RX pin of USB

Connect Tx0 of Serial0 of Arduino to TX of USB

- Then for the Vin pin of USB give 5v as input voltage and gnd the other pin.

Then the complete set up will be shown below

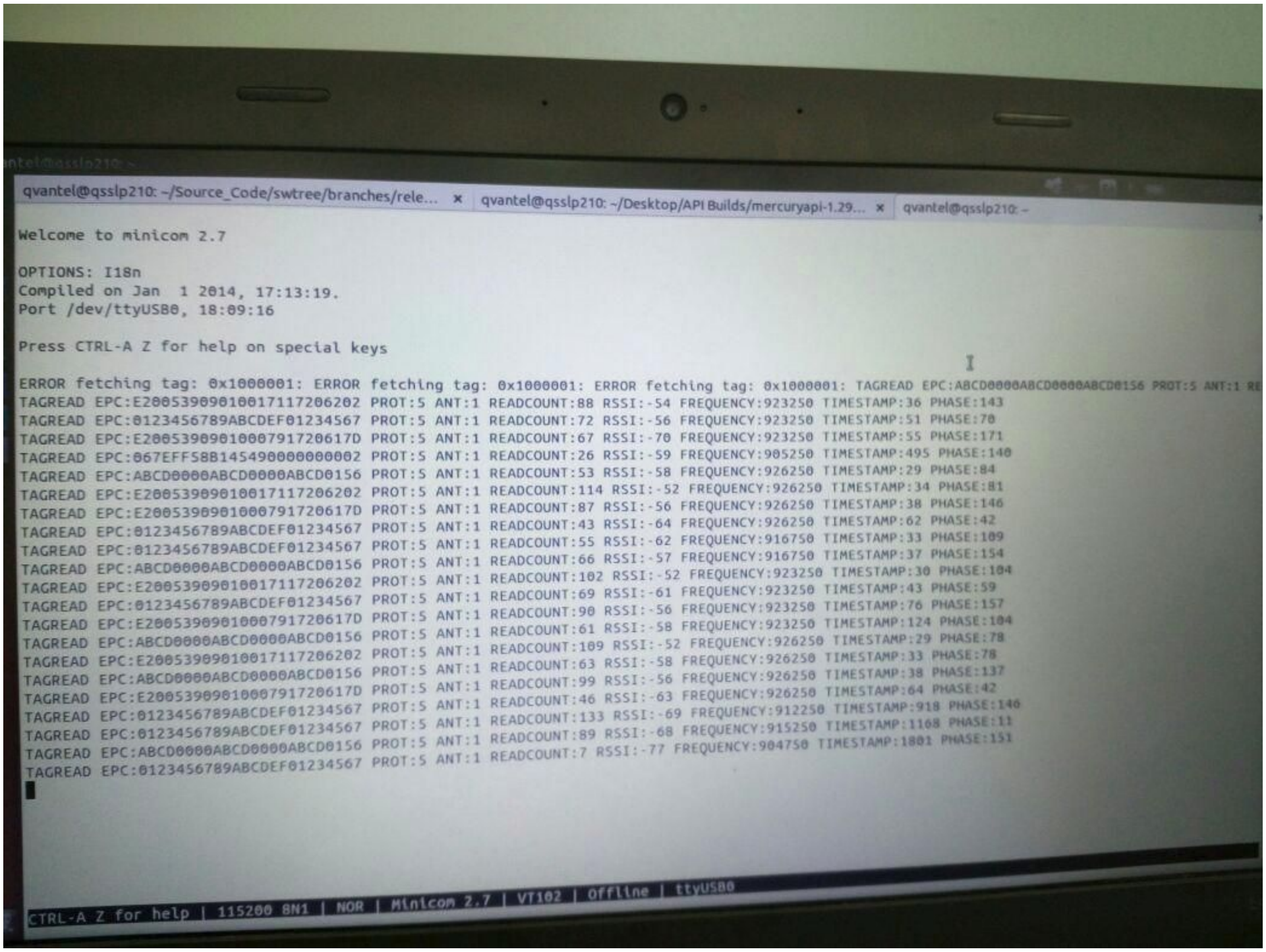


Test Results:

- To see the Tag read, open the Minicom in the command prompt and setting as follows:
    - Select the serial port.
    - Set the name of the USB as USBtty0.
    - Baud rate should be maintained as 115200.

Save it to dlf.

- When the tag is reading, user **“L”** led will blink . when transmitting, TX LED on the Arduino will blink.



qvantel@qsslp210: ~/Source_Code/swtree/branches/rele... ×
qvantel@qsslp210: ~/Desktop/API Builds/mercuryapi-1.29... ×
qvantel@qsslp210: ~

```
Welcome to minicom 2.7

OPTIONS: I18n
Compiled on Jan  1 2014, 17:13:19.
Port /dev/ttyUSB0, 18:09:16

Press CTRL-A Z for help on special keys

ERROR fetching tag: 0x1000001: ERROR fetching tag: 0x1000001: ERROR fetching tag: 0x1000001: TAGREAD EPC:ABCD0000ABCD0000ABCD0156 PROT:5 ANT:1 RE
TAGREAD EPC:E20053909010017117206202 PROT:5 ANT:1 READCOUNT:88 RSSI:-54 FREQUENCY:923250 TIMESTAMP:36 PHASE:143
TAGREAD EPC:0123456789ABCDEF01234567 PROT:5 ANT:1 READCOUNT:72 RSSI:-56 FREQUENCY:923250 TIMESTAMP:51 PHASE:70
TAGREAD EPC:E2005390901000791720617D PROT:5 ANT:1 READCOUNT:67 RSSI:-70 FREQUENCY:923250 TIMESTAMP:55 PHASE:171
TAGREAD EPC:067EFF58B145490000000002 PROT:5 ANT:1 READCOUNT:26 RSSI:-59 FREQUENCY:905250 TIMESTAMP:495 PHASE:140
TAGREAD EPC:ABCD0000ABCD0000ABCD0156 PROT:5 ANT:1 READCOUNT:53 RSSI:-58 FREQUENCY:926250 TIMESTAMP:29 PHASE:84
TAGREAD EPC:E20053909010017117206202 PROT:5 ANT:1 READCOUNT:114 RSSI:-52 FREQUENCY:926250 TIMESTAMP:34 PHASE:81
TAGREAD EPC:E2005390901000791720617D PROT:5 ANT:1 READCOUNT:87 RSSI:-56 FREQUENCY:926250 TIMESTAMP:38 PHASE:146
TAGREAD EPC:0123456789ABCDEF01234567 PROT:5 ANT:1 READCOUNT:43 RSSI:-64 FREQUENCY:926250 TIMESTAMP:62 PHASE:42
TAGREAD EPC:0123456789ABCDEF01234567 PROT:5 ANT:1 READCOUNT:55 RSSI:-62 FREQUENCY:916750 TIMESTAMP:33 PHASE:109
TAGREAD EPC:ABCD0000ABCD0000ABCD0156 PROT:5 ANT:1 READCOUNT:66 RSSI:-57 FREQUENCY:916750 TIMESTAMP:37 PHASE:154
TAGREAD EPC:E20053909010017117206202 PROT:5 ANT:1 READCOUNT:102 RSSI:-52 FREQUENCY:923250 TIMESTAMP:30 PHASE:104
TAGREAD EPC:0123456789ABCDEF01234567 PROT:5 ANT:1 READCOUNT:69 RSSI:-61 FREQUENCY:923250 TIMESTAMP:43 PHASE:59
TAGREAD EPC:E2005390901000791720617D PROT:5 ANT:1 READCOUNT:90 RSSI:-56 FREQUENCY:923250 TIMESTAMP:76 PHASE:157
TAGREAD EPC:ABCD0000ABCD0000ABCD0156 PROT:5 ANT:1 READCOUNT:61 RSSI:-58 FREQUENCY:923250 TIMESTAMP:124 PHASE:104
TAGREAD EPC:E20053909010017117206202 PROT:5 ANT:1 READCOUNT:109 RSSI:-52 FREQUENCY:926250 TIMESTAMP:29 PHASE:78
TAGREAD EPC:ABCD0000ABCD0000ABCD0156 PROT:5 ANT:1 READCOUNT:63 RSSI:-58 FREQUENCY:926250 TIMESTAMP:33 PHASE:78
TAGREAD EPC:E2005390901000791720617D PROT:5 ANT:1 READCOUNT:99 RSSI:-56 FREQUENCY:926250 TIMESTAMP:38 PHASE:137
TAGREAD EPC:0123456789ABCDEF01234567 PROT:5 ANT:1 READCOUNT:46 RSSI:-63 FREQUENCY:926250 TIMESTAMP:64 PHASE:42
TAGREAD EPC:0123456789ABCDEF01234567 PROT:5 ANT:1 READCOUNT:133 RSSI:-69 FREQUENCY:912250 TIMESTAMP:918 PHASE:140
TAGREAD EPC:ABCD0000ABCD0000ABCD0156 PROT:5 ANT:1 READCOUNT:89 RSSI:-68 FREQUENCY:915250 TIMESTAMP:1168 PHASE:11
TAGREAD EPC:0123456789ABCDEF01234567 PROT:5 ANT:1 READCOUNT:7 RSSI:-77 FREQUENCY:904750 TIMESTAMP:1801 PHASE:151

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7 | VT102 | Offline | ttyUSB0
```

***** while uploading the code remove the connections of SERIAL0 [RX0, TX0]. If those connections are not removed, then you will get AVRDUDE error”**

ThingMagic
A DIVISION OF TRIMBLE

Read.ino

```
#include <osdep.h>
#include <serial_reader_imp.h>
#include <tm_config.h>
#include <tm_reader.h>
#include <tmr_filter.h>
#include <tmr_gen2.h>
#include <tmr_gpio.h>
#include <tmr_ipx.h>
#include <tmr_iso180006b.h>
#include <tmr_params.h>
#include <tmr_read_plan.h>
#include <tmr_region.h>
#include <tmr_serial_reader.h>
#include <tmr_serial_transport.h>
#include <tmr_status.h>
#include <tmr_tag_auth.h>
#include <tmr_tag_data.h>
#include <tmr_tag_lock_action.h>
#include <tmr_tag_protocol.h>
#include <tmr_tagop.h>
#include <tmr_types.h>
#include <tmr_utils.h>

#define ERROR_BLINK_COUNT          10
#define ERROR_BLINK_INTERVAL   100
#define LOG_SERIAL_COMM    0

#define TAG_SEARCH_TIME     250
#define SEARCH_INTERVAL     1000

#define CONSOLE_BAUD_RATE          115200

#define TM_BAUD_RATE          115200
#define SERIAL_PROXY_BAUD_RATE 115200

#define SERIAL_PROXY_COM1          (&Serial1)

const int sysLedPin        = 13;

static int ledState          = LOW;   // ledState used to set the LED
static HardwareSerial *console = &Serial;
```

```
const int tmGpio1Pin             = 10;
const int tmGpio2Pin             = 9;
const int tmGpio3Pin             = 8;
const int tmGpio4Pin             = 7;

static TMR_Reader r, *rp;
static TMR_TransportListenerBlock tb;
static HardwareSerial *readerModule      = &Serial1;
static unsigned long nextTagSearchTime = 0;


static void  blink(int count, int blinkInterval)
{
  unsigned long blinkTime;
  unsigned long currentTime;
  blinkTime   = 0;
  currentTime = millis();
  while(count)
  {
          if(currentTime > blinkTime) {
          // save the last time you blinked the LED
          blinkTime = currentTime + blinkInterval;

          // if the LED is off turn it on and vice-versa:
          if (ledState == LOW)
          {
          ledState = HIGH;
          }
          else
          {
          ledState = LOW;
          --count;
          }
          // set the LED with the ledState of the variable:
          digitalWrite(sysLedPin, ledState);
          }
          currentTime = millis();
  }
}

static void checkerr(TMR_Reader* rp, TMR_Status ret, int exitval, const char *msg)
{
  if (TMR_SUCCESS != ret)
  {
          while ( 1 )
          {
          console->print("ERROR ");
          console->print(msg);
          console->print(": 0x");
          console->print(ret,HEX);
          console->print(": ");
          // console->println(TMR_strerror(ret));
          blink(ERROR_BLINK_COUNT,ERROR_BLINK_INTERVAL);
          }
  }
}

static void printComment(char *msg)
{
  console->print("#");
  console->println(msg);
}

static void initializeReader()
{
  TMR_Status ret;

  rp = &r;
  ret = TMR_create(rp, "tmr:///Serial1");
  checkerr(rp, ret, 1, "creating reader");

  {
          uint32_t rate;
          rate = TM_BAUD_RATE;
          ret = rp->paramSet(rp, TMR_PARAM_BAUDRATE, &rate);
  }

  // Set region to North America
  {
          TMR_Region region;
          region = TMR_REGION_NA;
          ret = TMR_paramSet(rp, TMR_PARAM_REGION_ID, &region);
          checkerr(rp, ret, 1, "setting region");
  }
  ret = TMR_connect(rp);
  checkerr(rp, ret, 1, "connecting reader");

  // The number of tags in the field at once is expected to be few, and we want to see those tags as much as
possible
  // So set the Session to 0
  {
        TMR_GEN2_Session value;

        value = TMR_GEN2_SESSION_S0;
        ret = TMR_paramSet(rp, TMR_PARAM_GEN2_SESSION, &value);
        checkerr(rp, ret, 1, "setting session");
  }

  // set the reader to toggle targets
  {
        TMR_GEN2_Target value;

        value = TMR_GEN2_TARGET_AB;
        ret = TMR_paramSet(rp, TMR_PARAM_GEN2_TARGET, &value);
        checkerr(rp, ret, 1, "setting target AB");
  }
}

static void reportTags()
{
  TMR_Status ret;

  while (TMR_SUCCESS == TMR_hasMoreTags(rp))
  {

        TMR_TagReadData trd;
        char epcStr[128];
        // blink(20,300);

        ret = TMR_getNextTag(rp, &trd);
        checkerr(rp, ret, 1, "fetching tag");

        TMR_bytesToHex(trd.tag.epc, trd.tag.epcByteCount, epcStr);
        console->print("TAGREAD EPC:");
        console->print(epcStr);

        if (TMR_TRD_METADATA_FLAG_PROTOCOL & trd.metadataFlags)
```

```
            {
            console->print(" PROT:");
            console->print(trd.tag.protocol);
            }
            if (TMR_TRD_METADATA_FLAG_ANTENNAID & trd.metadataFlags)
            {
            console->print(" ANT:");
            console->print(trd.antenna);
            }
            if (TMR_TRD_METADATA_FLAG_READCOUNT & trd.metadataFlags)
            {
            console->print(" READCOUNT:");
            console->print(trd.readCount);
            }
            if (TMR_TRD_METADATA_FLAG_RSSI & trd.metadataFlags)
            {
            console->print(" RSSI:");
            console->print(trd.rssi);
            }
            if (TMR_TRD_METADATA_FLAG_FREQUENCY & trd.metadataFlags)
            {
            console->print(" FREQUENCY:");
            console->print(trd.frequency);
            }
//          if (TMR_TRD_METADATA_FLAG_TIMESTAMP & trd.metadataFlags)
//          {
//          console->print(" TIMESTAMP:");
//          console->print(trd.dspMicros);
//          }
            if (TMR_TRD_METADATA_FLAG_PHASE & trd.metadataFlags)
            {
            console->print(" PHASE:");
            console->print(trd.phase);
            }
  }
 }

static void readTags()
{

  TMR_Status ret;

  ret = TMR_read(rp, TAG_SEARCH_TIME, NULL);
  checkerr(rp, ret, 1, "reading tags");

  reportTags();

}

void setup()
{
  // Set the GPIO pins that we are concerned about to output
  pinMode(sysLedPin,     OUTPUT);
  // initialize the GPIO pins
  digitalWrite(sysLedPin, 0);

  // start the console interface
  console->begin(CONSOLE_BAUD_RATE);

  initializeReader();
 console->print("\n");
}

void loop() {
  unsigned long currentTime;

  currentTime = millis();

  if(currentTime >= nextTagSearchTime)
  {
            // save the last time a search was performed
            nextTagSearchTime = currentTime + SEARCH_INTERVAL;
            readTags();
currentTime = millis();
  }
}
```

Read_nano.ino

```
#include <osdep.h>
#include <serial_reader_imp.h>
#include <tm_config.h>
#include <tm_reader.h>
#include <tmr_filter.h>
#include <tmr_gen2.h>
#include <tmr_gpio.h>
#include <tmr_ipx.h>
#include <tmr_iso180006b.h>
#include <tmr_params.h>
#include <tmr_read_plan.h>
#include <tmr_region.h>
#include <tmr_serial_reader.h>
#include <tmr_serial_transport.h>
#include <tmr_status.h>
#include <tmr_tag_auth.h>
#include <tmr_tag_data.h>
#include <tmr_tag_lock_action.h>
#include <tmr_tag_protocol.h>
#include <tmr_tagop.h>
#include <tmr_types.h>
#include <tmr_utils.h>

#define ERROR_BLINK_COUNT      10
#define ERROR_BLINK_INTERVAL   100
#define LOG_SERIAL_COMM        0

#define TAG_SEARCH_TIME        250
#define SEARCH_INTERVAL        1000

#define CONSOLE_BAUD_RATE      115200

#define TM_BAUD_RATE           115200
#define SERIAL_PROXY_BAUD_RATE 115200

#define SERIAL_PROXY_COM1      (&Serial1)

const int sysLedPin          = 13;
```

```
static int ledState          = LOW;   // ledState used to set the LED
static HardwareSerial *console = &Serial;

const int tmGpio1Pin         = 10;
const int tmGpio2Pin         = 9;
const int tmGpio3Pin         = 8;
const int tmGpio4Pin         = 7;

static TMR_Reader r, *rp;
static TMR_TransportListenerBlock tb;
static HardwareSerial *readerModule    = &Serial1;
static unsigned long nextTagSearchTime = 0;


static void  blink(int count, int blinkInterval)
{
  unsigned long blinkTime;
  unsigned long currentTime;
  blinkTime   = 0;
  currentTime = millis();
  while(count)
  {
    if(currentTime > blinkTime) {
      // save the last time you blinked the LED
      blinkTime = currentTime + blinkInterval;

      // if the LED is off turn it on and vice-versa:
      if (ledState == LOW)
      {
        ledState = HIGH;
      }
      else
      {
        ledState = LOW;
        --count;
      }
      // set the LED with the ledState of the variable:
      digitalWrite(sysLedPin, ledState);
    }
    currentTime = millis();
  }
}

static void checkerr(TMR_Reader* rp, TMR_Status ret, int exitval, const char *msg)
{
  if (TMR_SUCCESS != ret)
  {
    while ( 1 )
    {
      console->print("ERROR ");
      console->print(msg);
      console->print(": 0x");
      console->print(ret,HEX);
      console->print(": ");
     // console->println(TMR_strerror(ret));
      blink(ERROR_BLINK_COUNT,ERROR_BLINK_INTERVAL);
    }
  }
}

static void printComment(char *msg)
{
  console->print("#");
  console->println(msg);
}

static void initializeReader()
{
  TMR_Status ret;

  rp = &r;
  ret = TMR_create(rp, "tmr:///Serial1");
  checkerr(rp, ret, 1, "creating reader");

  {
    uint32_t rate;
    rate = TM_BAUD_RATE;
```

```
   ret = rp->paramSet(rp, TMR_PARAM_BAUDRATE, &rate);
  }

  // Set region to North America
  {
   TMR_Region region;
   region = TMR_REGION_NA3;
   ret = TMR_paramSet(rp, TMR_PARAM_REGION_ID, &region);
   checkerr(rp, ret, 1, "setting region");
  }

  ret = TMR_connect(rp);
  checkerr(rp, ret, 1, "connecting reader");

  // The number of tags in the field at once is expected to be few, and we want to see those tags as much
as possible
  // So set the Session to 0
  {
   TMR_GEN2_Session value;

   value = TMR_GEN2_SESSION_S0;
   ret = TMR_paramSet(rp, TMR_PARAM_GEN2_SESSION, &value);
   checkerr(rp, ret, 1, "setting session");
  }

  // set the reader to toggle targets
  {
   TMR_GEN2_Target value;

   value = TMR_GEN2_TARGET_AB;
   ret = TMR_paramSet(rp, TMR_PARAM_GEN2_TARGET, &value);
   checkerr(rp, ret, 1, "setting target AB");
  }
}

static void reportTags()
{
  TMR_Status ret;
```

```
while (TMR_SUCCESS == TMR_hasMoreTags(rp))
{

  TMR_TagReadData trd;
  char epcStr[128];
  // blink(20,300);
    console->print("\n");
  ret = TMR_getNextTag(rp, &trd);
  checkerr(rp, ret, 1, "fetching tag");

  TMR_bytesToHex(trd.tag.epc, trd.tag.epcByteCount, epcStr);
  console->print("TAGREAD EPC:");
  console->print(epcStr);

  if (TMR_TRD_METADATA_FLAG_PROTOCOL & trd.metadataFlags)
  {
    console->print(" PROT:");
    console->print(trd.tag.protocol);
  }
  if (TMR_TRD_METADATA_FLAG_ANTENNAID & trd.metadataFlags)
  {
    console->print(" ANT:");
    console->print(trd.antenna);
  }
  if (TMR_TRD_METADATA_FLAG_READCOUNT & trd.metadataFlags)
  {
    console->print(" READCOUNT:");
    console->print(trd.readCount);
  }
  if (TMR_TRD_METADATA_FLAG_RSSI & trd.metadataFlags)
  {
    console->print(" RSSI:");
    console->print(trd.rssi);
  }
  if (TMR_TRD_METADATA_FLAG_FREQUENCY & trd.metadataFlags)
  {
    console->print(" FREQUENCY:");
    console->print(trd.frequency);
  }
//    if (TMR_TRD_METADATA_FLAG_TIMESTAMP & trd.metadataFlags)
//    {
//      console->print(" TIMESTAMP:");
//      console->print(trd.dspMicros);
//    }
    if (TMR_TRD_METADATA_FLAG_PHASE & trd.metadataFlags)
    {
      console->print(" PHASE:");
      console->print(trd.phase);
     }
  }
 }

static void readTags()
{

  TMR_Status ret;

            TMR_ReadPlan plan;

            uint8_t antennaCount = 0x1;

            uint8_t antennaList[1]= {1};

             ret = TMR_RP_init_simple(&plan, antennaCount, &antennaList[0], TMR_TAG_PROTOCOL_GEN2,
1000);

            checkerr(rp, ret, 1, "initializing the read plan");

            /* Commit read plan */

            ret = TMR_paramSet(rp, TMR_PARAM_READ_PLAN, &plan);

            checkerr(rp, ret, 1, "setting read plan");

  ret = TMR_read(rp, TAG_SEARCH_TIME, NULL);
  checkerr(rp, ret, 1, "reading tags");

  reportTags();
```

```
}

void setup()
{
  // Set the GPIO pins that we are concerned about to output
  pinMode(sysLedPin,     OUTPUT);
  // initialize the GPIO pins
  digitalWrite(sysLedPin,     0);

  // start the console interface
  console->begin(CONSOLE_BAUD_RATE);

  initializeReader();
 console->print("\n");
}

void loop() {
  unsigned long currentTime;

  currentTime = millis();

  if(currentTime >= nextTagSearchTime)
  {
    // save the last time a search was performed
    nextTagSearchTime = currentTime + SEARCH_INTERVAL;
    readTags();
currentTime = millis();
  }
}
```