Internet of Things lab 3 Fall 2023

Group 2 : Avhishek Biswas, Anuruddha Ekanayake, Amlan Balabantaray, Shaswati Behera

Task 1 : Sensing

Requirements

1. Sense the internal temperature every second.
2. Implement a sliding window to calculate the average temperature over the last 5 second using a stack.

Development Plan:

a. Procedure of Solving the Problem

1. Install the temperature library. `#include <TemperatureZero.h>`
2. Import and initialize the temperature. `TempZero.init()`
3. Create a function **ReadTemp()** to read the temperature at every second.
4. Set a general clock `REG_GCLK_CLKCTRL` .
5. Set a timer **TC4** in Matched Freqeuncy mode and make it tick for every 1 second.
6. Create a flag variable that will add turn on to led the temperature to be read.
7. Inside the readtemp function, also create a global counter `reportCount` to count of number of readings.
8. When reportCount is equal to 5 we calculate the average and store it in the variable `avgTemperature` which is also defined globally.

b. Configuration Table

Requirement	Register Name	Register Function	Register Value
Clock Configuration	`REG_GCLK_CLKCTRL`	Configure Generic Clock Control	`GCLK_CLKCTRL_CLKEN
Timer Configuration	`TC->CTRLA.reg`	Timer Control A Register	`TC_CTRLA_MODE_COUNT16
Timer Frequency	`TC->CC[0].reg`	Timer Compare/Capture Register	`compareValue`
Timer Interrupt	`TC->INTENSET.reg`	Timer Interrupt Enable Set Register	`TC_INTENSET.bit.MC0 = 1`
NVIC Configuration	`NVIC_EnableIRQ()`	Nested Vector Interrupt Controller	`TC3_IRQn`

b. Run-time Errors

Error Code	Error
mem0	memory overflow
timer	timer counting error
flag0	seconds counting flag mismatch

2. Development Process:

Subtask 1:

```
void setup() {
  SerialUSB.begin(9600);
  TempZero.init();
  pinMode(PIN_LED_13, OUTPUT);
  startTimer(1);
}

float Readtemp() {
  if (canReadTemp) {
    float temperature = TempZero.readInternalTemperature();
    SerialUSB.print("Internal Temperature is: ");
    SerialUSB.println(temperature);
    }
```

Subtask 2:

```
float Readtemp() {
  if (canReadTemp) {
    float temperature = TempZero.readInternalTemperature();
    SerialUSB.print("Internal Temperature is: ");
    SerialUSB.println(temperature);
    // Remove the oldest temperature from the sum
    tempSum -= tempBuffer[tempIndex];

    // Add the new temperature to the buffer and sum
    tempBuffer[tempIndex] = temperature;
    tempSum += temperature;

    // Update the index for the oldest temperature
    tempIndex = (tempIndex + 1) % WINDOW_SIZE;

    // Update the count of temperatures added to the buffer
    if (tempCount < WINDOW_SIZE) {
      tempCount++;
    }

    // Increment the report counter
    reportCount++;

    // If 5 readings have been taken, calculate the average temperature
    if (reportCount >= WINDOW_SIZE) {
      avgTemperature = tempSum / tempCount;
      newAvgAvailable = true;  // Set the flag
      reportCount = 0;  // Reset the report counter
    }

    canReadTemp = false;
  }
  return avgTemperature;
}
```

3. Test Plan:

1. **LED Blinking**:
   - Verify that the Blue LED toggles on and off at an interval of 500 milliseconds.
   - Verify that the Yellow LED toggles on and off at an interval of 1000 milliseconds.
2. **Serial Output**:
   - Verify that the Serial Monitor displays the correct LED status messages ("Blue LED is on/off", "Yellow LED is on/off") in real-time.

Component	Test Description	Result	Comment
Blue LED	Toggle on/off at 1s interval	Pass	LED toggled as expected at 1-second intervals

Component	Test Description	Result	Comment
Serial Output	Display "Blue LED is on/off"	Pass	Messages displayed correctly in the serial monitor
Record Temp at 1sec	Display "Internal Temperature is: "	Pass	Messages displayed correctly in the serial monitor
Record Average Temp at 5sec	Display "Average Temperature over last 5 seconds is:"	Pass	Messages displayed correctly in the serial monitor
System Level	Blue LEDs toggle at correct intervals	Pass	Both LEDs toggled at their respective intervals without conflict

Screenshot

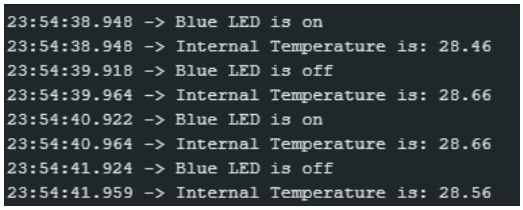

Fig.1 - Output showing temperature being recorded and reported every second,with LED turned on and off for each second.

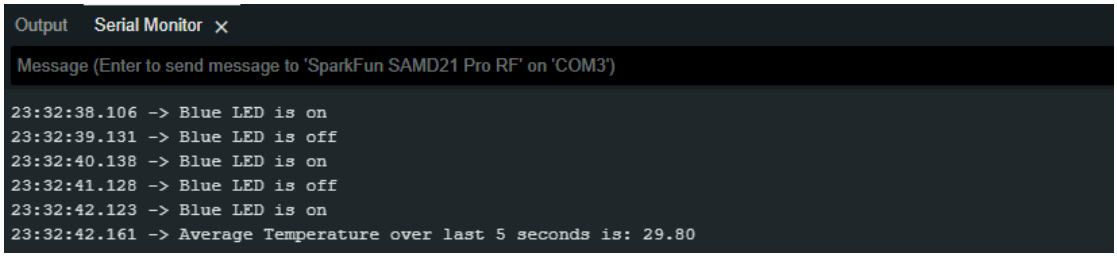

Fig.2 - Output showing temperature being recorded and reported every 5 seconds,with LED turned on and off for each second.

Task 2 : Communication

Requirements

1.Packet structure: Design a packet structure including the following information

1. Node ID
2. Packet ID
3. Timestamp
4. Payload: sensor data, error log data
5. Communicate with your teammate's board and transmit your temperature data every 5 second such that every board holds a copy of the average temperature of all the nodes, i.e.(Alice, temperature value), (Bob, temperature value), (Carter, temperature value)
6. Elect a node as the leader in the network and send error log to that board
7. Ensure that there is a mechanism to avoid collision.

Development Plan:

a. Procedure of Solving the Problem

b. Configuration Table

b. Run-time Errors

Error Code	Error
mem0	memory overflow
timer	timer counting error
flag0	seconds counting flag mismatch

2. Development Process:

a. Method of Problem Solving:

1. We choose the Polling method as our way to implement the communication between nodes.
2. Therefore, we add another information called `authID` which equals to `2` for `Child Node 2`. `3` for `Child Node 3`, `4` for `Child Node 4`.
3. At the Master Node:
4. We use the `reportCount` which is a timer controlled variable to talk to each Child Node at a perticular second. The value of this variable increases from 0 to 4 for each second counted by the timer.
5. As the `reportCount` value becomes 2, we create a packet and send it to Node 2 with `authID = 2` using `rf95.send(toSend, sizeof(Packet));`
6. Then we switch to recieving mode using `rf95.available()`.
7. We then wait for the transmission to complete using a while loop.
8. As the Master Node recieves a packet from the perticular Node, it processes the packet, and saves the payload and the error values, which if present is added to the flash storage.
9. This process is repeated for every child Node 3 and 4 respectively.
10. After every transmission for each Node we print the payload values for each Node that is recorded and the errors if present and the FLash storage content.
11. We also introduce Nodeflag variables, which are basically used so that in the same second the Master does not send multiple packets to a child Node requesting it's payload.
12. Also, something to be noted, at the first transmission the Node 1 will not have an average temperature as 5 secs has not passed therefore all the other nodes will be noting it's temperature to be 0. And from the second transmission the average temperature will be showing up this is demostrated with the screenshots of Fig. 3 and 4.
13. At the Child Nodes:
    1. These Node start their functioning by listening into the channel.
    2. As soon as they receive a packet, they open it and check for the `Node ID` and the `Auth ID`.
        1. If the `Node ID = 1` and the `Auth ID = ChildNode_ID` then it is a request from the Master Node for the child Nodes packet.
        2. And the child Node transmits it's packet with the payload and errors if any.
        3. If it is not the Master Node, the child Node with open the packet and record the `Node ID` and the `payload` which will stored in other Nodes temperature variable.
        4. After every received Node the child Nodes print the latest contents of each Nodes in the SerialUSB.

b. Run-time Errors

Error Code	Error
mem0	memory overflow
timer	timer counting error
flag0	seconds counting flag mismatch
WDT	Watchdog timer error due to reset

Error Code	Error
packet	no recieved packet from the child Node
infiniteloop	when no packet is received the Master is stuck in an infinite loop

Subtask 1:

Packet Structure - Along with our 4 given information, we decided to add an authID space in the packet that will help in the child nodes to identify who needs to transmit. We define this packet as a C++ structure and initialize them.

```
struct Packet {
  uint8_t nodeID;
  uint16_t packetID;
  uint32_t timestamp;
  float payload;
  char error[16];
  uint8_t authID;

  // Constructor to initialize the values
  Packet() : nodeID(0), packetID(0), timestamp(0), payload(0.0), authID(0) {
    // Initialize the error array to zeros
    memset(error, '0', sizeof(error));
  }
};
```

Subtask 2:

Communicate with teammates board to print eachother's temperature - In the below code we define the node temperature with for each board and we set them based in an ifelse blocks based on the recieved packet Node ID. And we print that temperature after every recieved packet.

```
float node1_temperature = 0;
float node2_temperature = 0;
float node3_temperature = 0;
float node4_temperature = 0;


 // SAVE NODE 1 'S TEMPERATURE
 node1_temperature = receivedPacket->payload;

 // SAVING NODE 2 AND 3 TEMPERATURES
if(receivedPacket->nodeID == 2){
 // SAVE NODE 2 'S TEMPERATURE
 node2_temperature = receivedPacket->payload;
}
else if(receivedPacket->nodeID == 3){
 // SAVE NODE 3 'S TEMPERATURE
 node3_temperature = receivedPacket->payload;
}

node4_temperature = avgTemperature;

SerialUSB.println();
SerialUSB.println();

SerialUSB.println("PRINTING EVERYONE'S TEMPERATURE");
SerialUSB.print("(Avhi, "); SerialUSB.print(node1_temperature, 2);
SerialUSB.print("), (Amlan, "); SerialUSB.print(node2_temperature, 2);
SerialUSB.print("), (Shaswati, "); SerialUSB.print(node3_temperature, 2);
SerialUSB.print("), (Anu, "); SerialUSB.print(node4_temperature, 2);
SerialUSB.println(")");

SerialUSB.println();
SerialUSB.println();
```

Subtask 3:

Elect a leader Node - Node Name Avhi with id 1 is selected as the leader/master for the communication structure.

```
// Master sends it average temperature
  else if(reportCount == 4 && Master == false)
  {  // set the slaves false for the next 5 secs
      Slave2 = false;
      Slave3 = false;
      Slave4 = false;
      Master = true;
      // Send the authentication packet
      SerialUSB.println("Sending Master's Average Temperature. ");

      //SAVE THE TEMPERATURE OF NODE 2
      node1_temperature = avgTemperature;

      // Create the packet
      Packet packet;
      packet.nodeID = 1;  // Node 1
      packet.packetID = packetCounter++;
      packet.timestamp = millis();  // Current time in milliseconds
      packet.payload = avgTemperature;
      // packet.error = 0;
      packet.authID = 10;  // master is sending it's temperature

      // Serialize the packet into a byte array
      uint8_t toSend[sizeof(Packet)];
      memcpy(toSend, &packet, sizeof(Packet));

      // Print and send the message
      SerialUSB.print("SENDING TEMPERATURE OF MASTER TO ALL NODES : PACKET NUMBER :");
      SerialUSB.println(packet.packetID);
      SerialUSB.println();
      SerialUSB.println();
      rf95.send(toSend, sizeof(Packet));

      SerialUSB.println();
      SerialUSB.println();

      SerialUSB.println("PRINTING EVERYONE'S TEMPERATURE");
      SerialUSB.print("(Avhi, "); SerialUSB.print(node1_temperature, 2);
      SerialUSB.print("), (Amlan, "); SerialUSB.print(node2_temperature, 2);
      SerialUSB.print("), (Shaswati, "); SerialUSB.print(node3_temperature, 2);
      SerialUSB.print("), (Anu, "); SerialUSB.print(node4_temperature, 2);
      SerialUSB.println(")");

      SerialUSB.println();
      SerialUSB.println();
  }
```

Subtask 4:

Collision Avoidance - For Collision avoidance we select a polling method where the Master Node or Node ID - 1, with name Avhi, reaches out to each node by changing the authentication ID in the packet, and asks for their temperature packet. Upon recieving a request, every child node immidiately gathers all the parameters in the packet and sends the packet over the air. During this time all other nodes are listing and are checking continuously if the Master Node is transmitting and if so, is the authentication ID matches their node.

The below code is from the void loop, where we implement the collision avoidance algorithm, for different nodes.

```
void loop()
{ avgTemperature = Readtemp();
  // for Node =  slave 2
  if(reportCount == 1 && Slave2 == false)
  {    Master = false;
       Slave2 = true;
      // Send the authentication packet
      SerialUSB.println("Asking Node 2 to send it's temp. ");

      // Create the packet
      Packet packet;
      packet.nodeID = 1;  // Node 1
      packet.packetID = packetCounter++;
      packet.timestamp = millis();  // Current time in milliseconds
      packet.payload = avgTemperature;
      //packet.error = 0;
      packet.authID = 2;

      // Serialize the packet into a byte array
      uint8_t toSend[sizeof(Packet)];
      memcpy(toSend, &packet, sizeof(Packet));

      // Print and send the message
      SerialUSB.print("Sending authentication packet to Node 2: Packet Number :");
      SerialUSB.println(packet.packetID);
      rf95.send(toSend, sizeof(Packet));

      // Recieve the data packet from Node 2
      // SerialUSB.println(rf95.available());
      while(rf95.available() == 0){}

    // Recieve the data packet from Node 2
      if (rf95.available()){
        // Should be a message for us now
        uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
        uint8_t len = sizeof(buf);
        if (rf95.recv(buf, &len))
        {
            // Cast the received buffer to the Packet struct type
            Packet *receivedPacket = (Packet *)buf;
            buf[len] = '\0';  // Null-terminate the received string
            digitalWrite(LED, HIGH); //Turn on status LED
            timeSinceLastPacket = millis(); //Timestamp this packet

            //SAVE THE TEMPERATURE OF NODE 2
            node2_temperature = receivedPacket->payload;
            node1_temperature = avgTemperature;

            // Print the received packet details
            SerialUSB.print("Got message from Node ID: ");
            SerialUSB.print(receivedPacket->nodeID);
            SerialUSB.print(", Packet ID: ");
            SerialUSB.print(receivedPacket->packetID);
            SerialUSB.print(", Timestamp: ");
            SerialUSB.print(receivedPacket->timestamp);
            SerialUSB.print(", Payload (Temperature): ");
            SerialUSB.print(receivedPacket->payload);

            SerialUSB.println();
            print_errors(receivedPacket->error);
            //FLASH STORAGE DETAILS PRINT
            SerialUSB.println("FLASH STORAGE RESULTS");
            SerialUSB.print("AMLAN: ");SerialUSB.print(error_Amlan.read());
            SerialUSB.print(", SHASWATI: ");SerialUSB.print(error_Shaswati.read());
            SerialUSB.print(", ANURUDDHA: ");SerialUSB.print(error_Anuruddha.read());
            SerialUSB.print(", AVHISHEK: ");SerialUSB.print(error_Avhishek.read());
            SerialUSB.print(", ERROR RECEIVE ");SerialUSB.println(error_receive.read());

            SerialUSB.println();
            SerialUSB.println();

            SerialUSB.println("PRINTING EVERYONE'S TEMPERATURE");
            SerialUSB.print("(Avhi, "); SerialUSB.print(node1_temperature, 2);
            SerialUSB.print("), (Amlan, "); SerialUSB.print(node2_temperature, 2);
            SerialUSB.print("), (Shaswati, "); SerialUSB.print(node3_temperature, 2);
```

```
            SerialUSB.print("), (Anu, "); SerialUSB.print(node4_temperature, 2);
            SerialUSB.println(")");

            SerialUSB.println();
            SerialUSB.println();


            write_error(receivedPacket->nodeID, receivedPacket->error) ;                      //Write the received packet error in the correct

               current_packet_id = receivedPacket->packetID;                //Assign the current packet ID to the current_packet_id variable

                     if(current_packet_id-previouse_packet_id > 1) {
                       receivedPacket->error[11]='1';
                     }           //check the missing packet
                    //  write_error(receivedPacket->nodeID, 19) ;
                     previouse_packet_id = current_packet_id;
          }
          else
          {
          SerialUSB.println("Recieve failed");
          char temp1[]={'0','0','0','0','0','0','0','0','0','0','0','0','1','0','0','0'};
          write_error(5, temp1);                                // Save the error code as message receive failed
          }
      }
      timeSinceLastPacket = millis();
}

// for Node = slave 3
else if(reportCount == 2 && Slave3 == false)
{   Slave3 = true;
    // Send the authentication packet
    SerialUSB.println("Asking Node 3 to send it's temp. ");

    // Create the packet
    Packet packet;
    packet.nodeID = 1;  // Node 1
    packet.packetID = packetCounter++;
    packet.timestamp = millis();  // Current time in milliseconds
    packet.payload = avgTemperature;
    // packet.error = 0;
    packet.authID = 3;

    // Serialize the packet into a byte array
    uint8_t toSend[sizeof(Packet)];
    memcpy(toSend, &packet, sizeof(Packet));

    // Print and send the message
    SerialUSB.print("Sending authentication packet to Node 3 : Packet Number : ");
    SerialUSB.println(packet.packetID);
    rf95.send(toSend, sizeof(Packet));

    // Recieve the data packet from Node 3
    // SerialUSB.println(rf95.available());
    while(rf95.available() == 0){
    }

  // Recieve the data packet from Node 3
    if (rf95.available()){
      // Should be a message for us now
      uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
      uint8_t len = sizeof(buf);
      if (rf95.recv(buf, &len))
      {
          // Cast the received buffer to the Packet struct type
          Packet *receivedPacket = (Packet *)buf;
          buf[len] = '\0';  // Null-terminate the received string
          digitalWrite(LED, HIGH); //Turn on status LED
          timeSinceLastPacket = millis(); //Timestamp this packet

          //SAVE THE TEMPERATURE OF NODE 2
          node3_temperature = receivedPacket->payload;
          node1_temperature = avgTemperature;

          // Print the received packet details
          SerialUSB.print("Got message from Node ID: ");
          SerialUSB.print(receivedPacket->nodeID);
```

```
          SerialUSB.print(", Packet ID: ");
          SerialUSB.print(receivedPacket->packetID);
          SerialUSB.print(", Timestamp: ");
          SerialUSB.print(receivedPacket->timestamp);
          SerialUSB.print(", Payload (Temperature): ");
          SerialUSB.print(receivedPacket->payload);
      //     SerialUSB.print(" RSSI: ");
      //     SerialUSB.print(rf95.lastRssi(), DEC);
          SerialUSB.println();
           print_errors(receivedPacket->error);
          //FLASH STORAGE DETAILS PRINT
          SerialUSB.println("FLASH STORAGE RESULTS");
          SerialUSB.print("AMLAN: ");SerialUSB.print(error_Amlan.read());
          SerialUSB.print(", SHASWATI: ");SerialUSB.print(error_Shaswati.read());
          SerialUSB.print(", ANURUDDHA: ");SerialUSB.print(error_Anuruddha.read());
          SerialUSB.print(", AVHISHEK: ");SerialUSB.print(error_Avhishek.read());
          SerialUSB.print(", ERROR RECEIVE ");SerialUSB.println(error_receive.read());


          SerialUSB.println();
          SerialUSB.println();

          SerialUSB.println("PRINTING EVERYONE'S TEMPERATURE");
          SerialUSB.print("(Avhi, "); SerialUSB.print(node1_temperature, 2);
          SerialUSB.print("), (Amlan, "); SerialUSB.print(node2_temperature, 2);
          SerialUSB.print("), (Shaswati, "); SerialUSB.print(node3_temperature, 2);
          SerialUSB.print("), (Anu, "); SerialUSB.print(node4_temperature, 2);
          SerialUSB.println(")");

          SerialUSB.println();
          SerialUSB.println();

          write_error(receivedPacket->nodeID, receivedPacket->error) ;                      //Write the received packet error in the correct

          current_packet_id = receivedPacket->packetID;                //Assign the current packet ID to the current_packet_id variable

          if(current_packet_id-previouse_packet_id > 1) {
            receivedPacket->error[11]='1';
          }
         // check the missing packet
         // write_error(receivedPacket->nodeID, 19) ;
           previouse_packet_id = current_packet_id;
         }
        else
        {
        SerialUSB.println("Recieve failed");
        char temp1[]={'0','0','0','0','0','0','0','0','0','0','0','0','1','0','0','0'};
        write_error(5, temp1);                                // Save the error code as message receive failed
        }
    }
    timeSinceLastPacket = millis();
}

// for Node = slave 4
else if(reportCount == 3 && Slave4 == false)
  {   Slave4 = true; Master = false;
     // Send the authentication packet
     SerialUSB.println("Asking Node 4 to send it's temp. ");

     // Create the packet
     Packet packet;
     packet.nodeID = 1;  // Node 1
     packet.packetID = packetCounter++;
     packet.timestamp = millis();  // Current time in milliseconds
     packet.payload = avgTemperature;
     // packet.error = 0;
     packet.authID = 4;

     // Serialize the packet into a byte array
     uint8_t toSend[sizeof(Packet)];
     memcpy(toSend, &packet, sizeof(Packet));

     // Print and send the message
     SerialUSB.print("Sending authentication packet to Node 4 : Packet Number : ");
     SerialUSB.println(packet.packetID);
     rf95.send(toSend, sizeof(Packet));
```

```
 // Recieve the data packet from Node 4
 // SerialUSB.println(rf95.available());
 while(rf95.available() == 0){}
 // SerialUSB.println(rf95.available());

 if (rf95.available()){
   // Should be a message for us now
   uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
   uint8_t len = sizeof(buf);
   if (rf95.recv(buf, &len))
   {
       // Cast the received buffer to the Packet struct type
       Packet *receivedPacket = (Packet *)buf;
       buf[len] = '\0';  // Null-terminate the received string
       digitalWrite(LED, HIGH); //Turn on status LED
       timeSinceLastPacket = millis(); //Timestamp this packet

       //SAVE THE TEMPERATURE OF NODE 2
       node4_temperature = receivedPacket->payload;
       node1_temperature = avgTemperature;

       // Print the received packet details
       SerialUSB.print("Got message from Node ID: ");
       SerialUSB.print(receivedPacket->nodeID);
       SerialUSB.print(", Packet ID: ");
       SerialUSB.print(receivedPacket->packetID);
       SerialUSB.print(", Timestamp: ");
       SerialUSB.print(receivedPacket->timestamp);
       SerialUSB.print(", Payload (Temperature): ");
       SerialUSB.print(receivedPacket->payload);
   //     SerialUSB.print(" RSSI: ");
   //     SerialUSB.print(rf95.lastRssi(), DEC);
       SerialUSB.println();
        print_errors(receivedPacket->error);
       //FLASH STORAGE DETAILS PRINT
       SerialUSB.println("FLASH STORAGE RESULTS");
       SerialUSB.print("AMLAN: ");SerialUSB.print(error_Amlan.read());
       SerialUSB.print(", SHASWATI: ");SerialUSB.print(error_Shaswati.read());
       SerialUSB.print(", ANURUDDHA: ");SerialUSB.print(error_Anuruddha.read());
       SerialUSB.print(", AVHISHEK: ");SerialUSB.print(error_Avhishek.read());
       SerialUSB.print(", ERROR RECEIVE ");SerialUSB.println(error_receive.read());


       SerialUSB.println();
       SerialUSB.println();

       SerialUSB.println("PRINTING EVERYONE'S TEMPERATURE");
       SerialUSB.print("(Avhi, "); SerialUSB.print(node1_temperature, 2);
       SerialUSB.print("), (Amlan, "); SerialUSB.print(node2_temperature, 2);
       SerialUSB.print("), (Shaswati, "); SerialUSB.print(node3_temperature, 2);
       SerialUSB.print("), (Anu, "); SerialUSB.print(node4_temperature, 2);
       SerialUSB.println(")");

       SerialUSB.println();
       SerialUSB.println();


       write_error(receivedPacket->nodeID, receivedPacket->error) ;                      //Write the received packet error in the correct

          current_packet_id = receivedPacket->packetID;                //Assign the current packet ID to the current_packet_id variable

                if(current_packet_id-previouse_packet_id > 1) {
                  receivedPacket->error[11]='1';
                }           //check the missing packet
               //  write_error(receivedPacket->nodeID, 19) ;
                previouse_packet_id = current_packet_id;
    }
    else
    {
    SerialUSB.println("Recieve failed");
    char temp1[]={'0','0','0','0','0','0','0','0','0','0','0','0','1','0','0','0'};
    write_error(5, temp1);                              // Save the error code as message receive failed
    }
}
```

```
      timeSinceLastPacket = millis();
  }

  // Master sends it average temperature
  else if(reportCount == 4 && Master == false)
  {  // set the slaves false for the next 5 secs
      Slave2 = false;
      Slave3 = false;
      Slave4 = false;
      Master = true;
      // Send the authentication packet
      SerialUSB.println("Sending Master's Average Temperature. ");

      //SAVE THE TEMPERATURE OF NODE 2
      node1_temperature = avgTemperature;

      // Create the packet
      Packet packet;
      packet.nodeID = 1;  // Node 1
      packet.packetID = packetCounter++;
      packet.timestamp = millis();  // Current time in milliseconds
      packet.payload = avgTemperature;
      // packet.error = 0;
      packet.authID = 10;  // master is sending it's temperature

      // Serialize the packet into a byte array
      uint8_t toSend[sizeof(Packet)];
      memcpy(toSend, &packet, sizeof(Packet));

      // Print and send the message
      SerialUSB.print("SENDING TEMPERATURE OF MASTER TO ALL NODES : PACKET NUMBER :");
      SerialUSB.println(packet.packetID);
      SerialUSB.println();
      SerialUSB.println();
      rf95.send(toSend, sizeof(Packet));

      SerialUSB.println();
      SerialUSB.println();

      SerialUSB.println("PRINTING EVERYONE'S TEMPERATURE");
      SerialUSB.print("(Avhi, "); SerialUSB.print(node1_temperature, 2);
      SerialUSB.print("), (Amlan, "); SerialUSB.print(node2_temperature, 2);
      SerialUSB.print("), (Shaswati, "); SerialUSB.print(node3_temperature, 2);
      SerialUSB.print("), (Anu, "); SerialUSB.print(node4_temperature, 2);
      SerialUSB.println(")");

      SerialUSB.println();
      SerialUSB.println();


  }
}
```

3. Test Plan:

Component	Test Description	Result	Comment
Blue LED	Toggle on/off at 1s interval	Pass	LED toggled as expected at 1-second intervals
Yellow LED	Toggle on/off at 1s interval	Pass	LED toggled as expected during reciving
Green LED	Toggle on/off at 1s interval	Pass	LED toggled as expected during transmitting
Serial Output	Showing average Temperature	Pass	Messages displayed correctly in the serial monitor
Serial Output	Showing Flash Memory Content at Master	Pass	Messages displayed correctly in the serial monitor
Serial Output	Showing Temperature for all Nodes at Master	Pass	Messages displayed correctly in the serial monitor

Component	Test Description	Result	Comment
Serial Output	Showing Temperature for all Nodes at Child Node	Pass	Messages displayed correctly in the serial monitor
Record Temp at 1sec	Display "Internal Temperature is: "	Pass	Messages displayed correctly in the serial monitor
Record Average Temp at 5sec	Display "Average Temperature over last 5 seconds is:"	Pass	Messages displayed correctly in the serial monitor
System Level	LEDs toggle at correct intervals	Pass	All LEDs toggled at their respective intervals without conflict

Screenshot

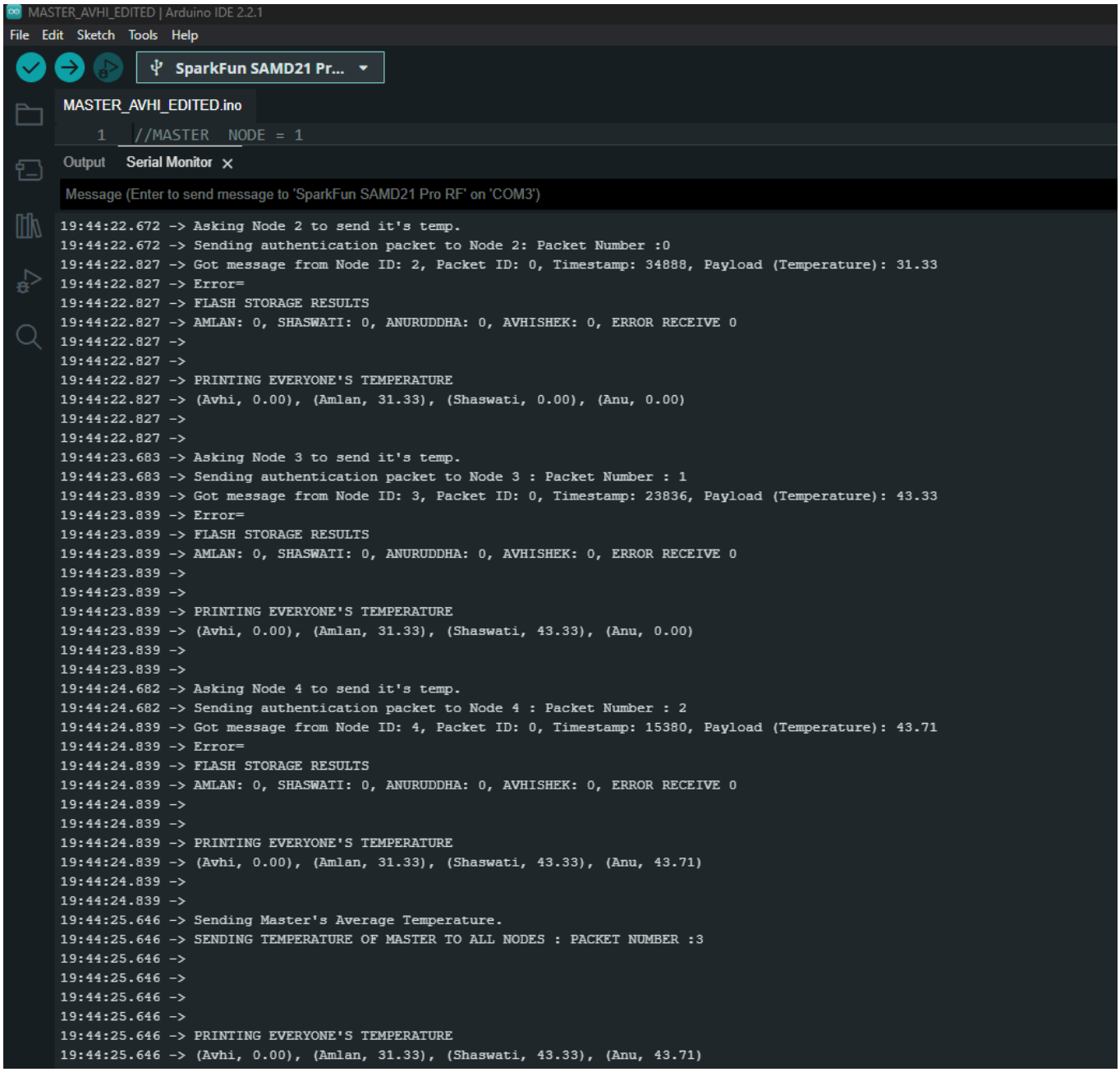

Fig 3. Master output serial log showing first communication with Node 2, Node 3 and Node 4. Then the Master prints the Flash Memory and the all the other Nodes temperature. We can see the temperature for Master being 0 for the first transmission.

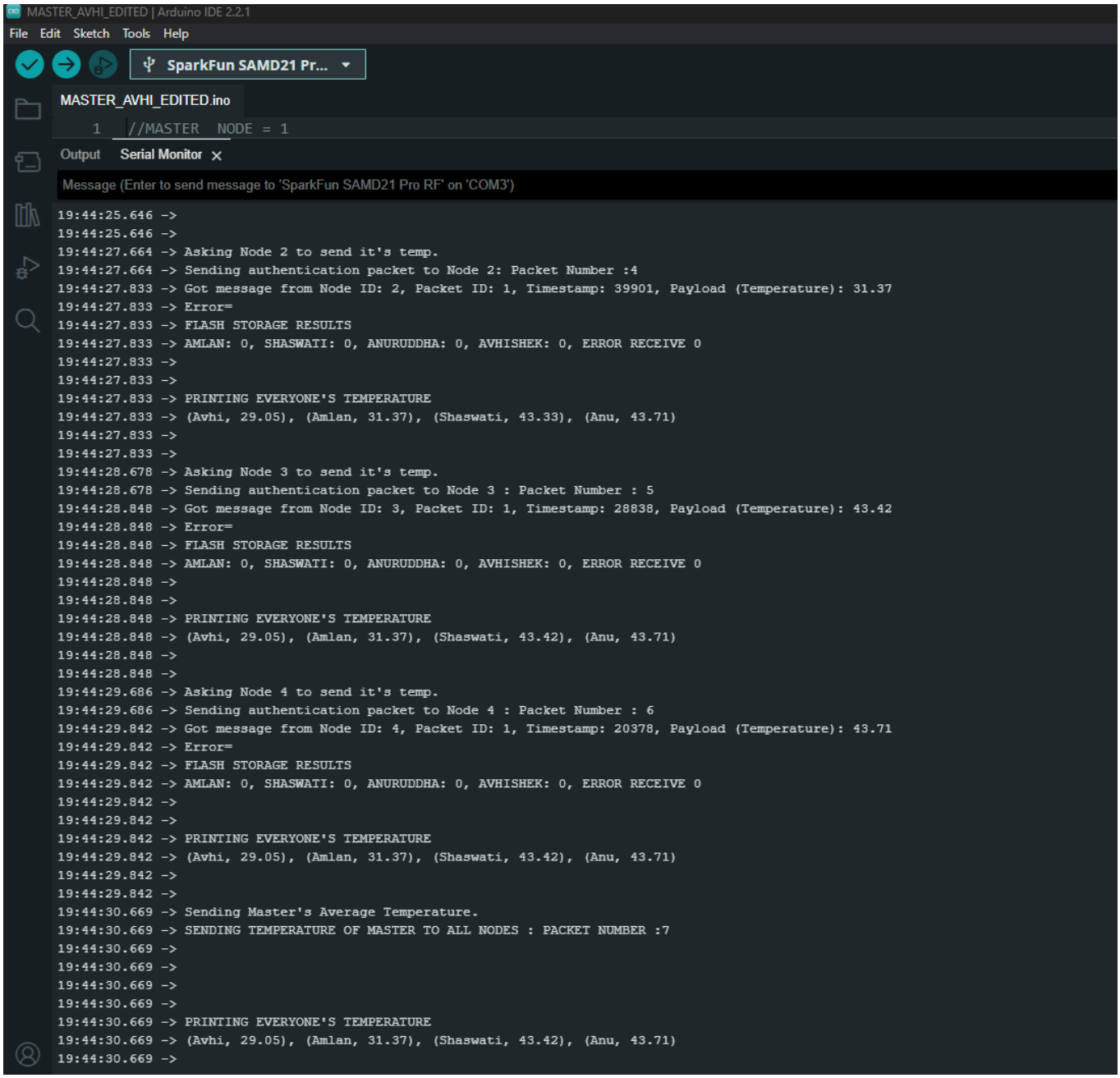

Fig 4. Master output serial log during second transmission with average calculated temperature value.

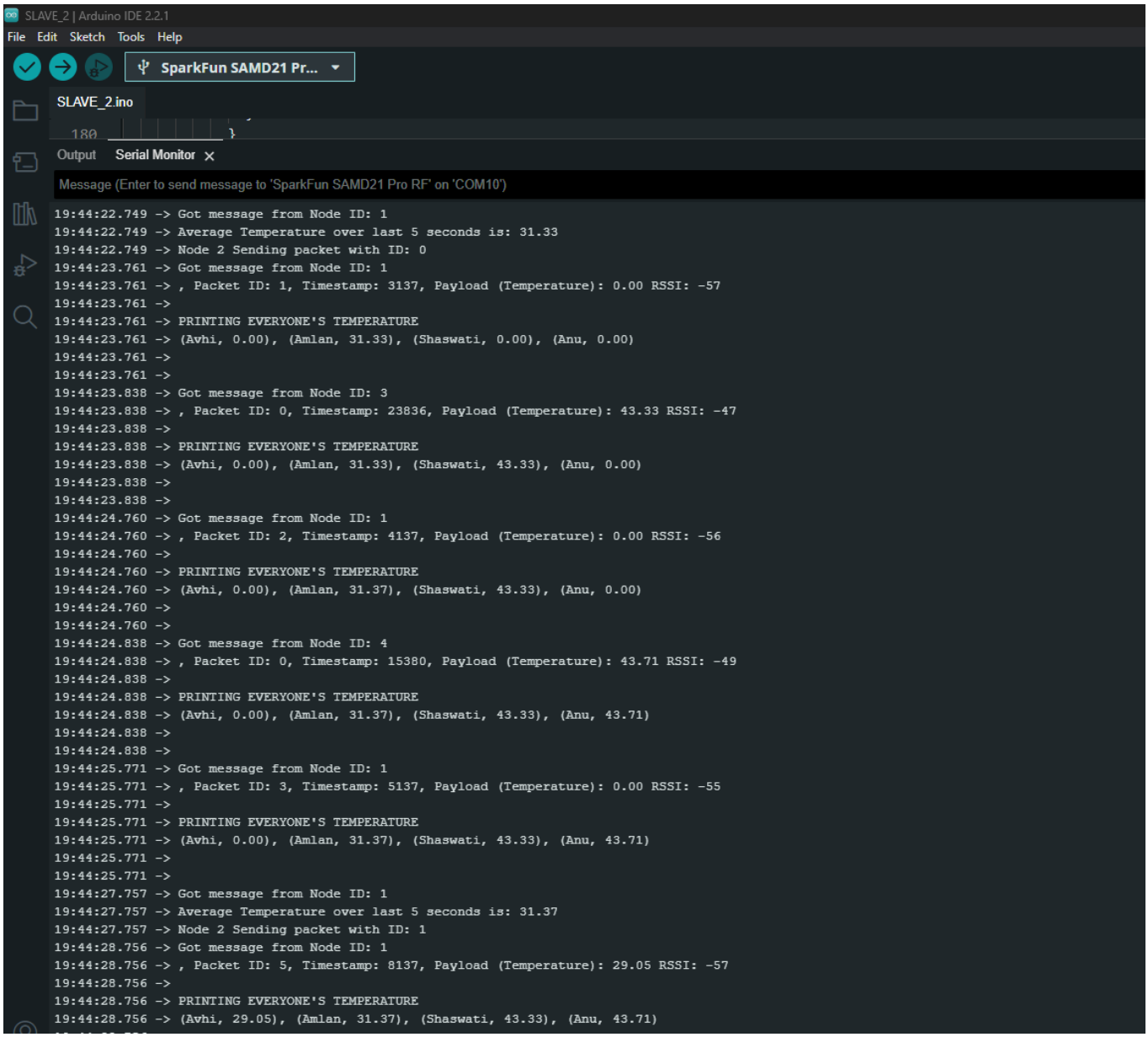

Fig 5. Child Node 2, showing authentication and packet transmission request from Master/Node1. Also subsequent logs show the temperature from other Nodes in required format.

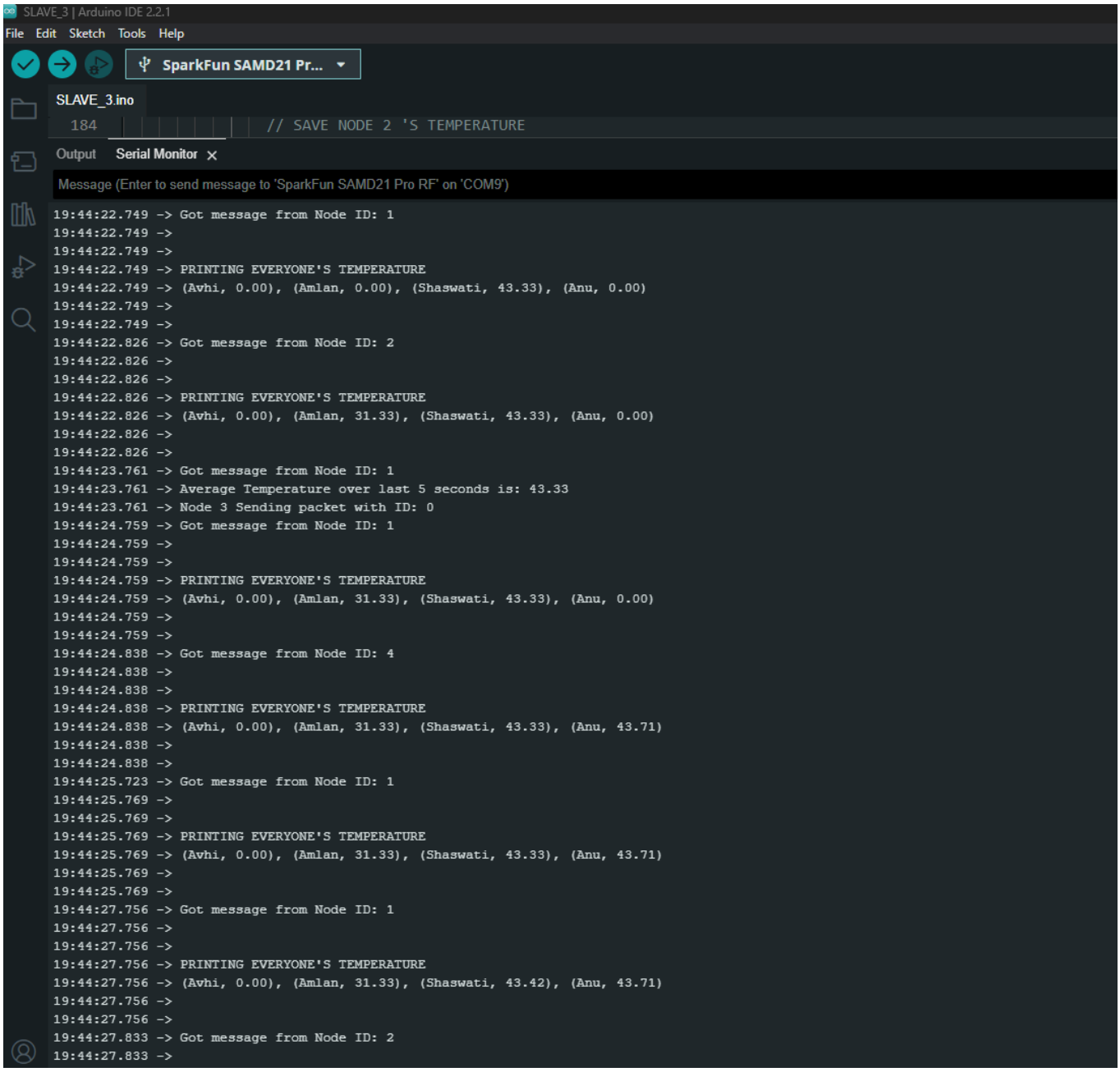

Fig 6. Child Node 3, showing authentication and packet transmission request from Master/Node1. Also subsequent logs show the temperature from other Nodes in required format.

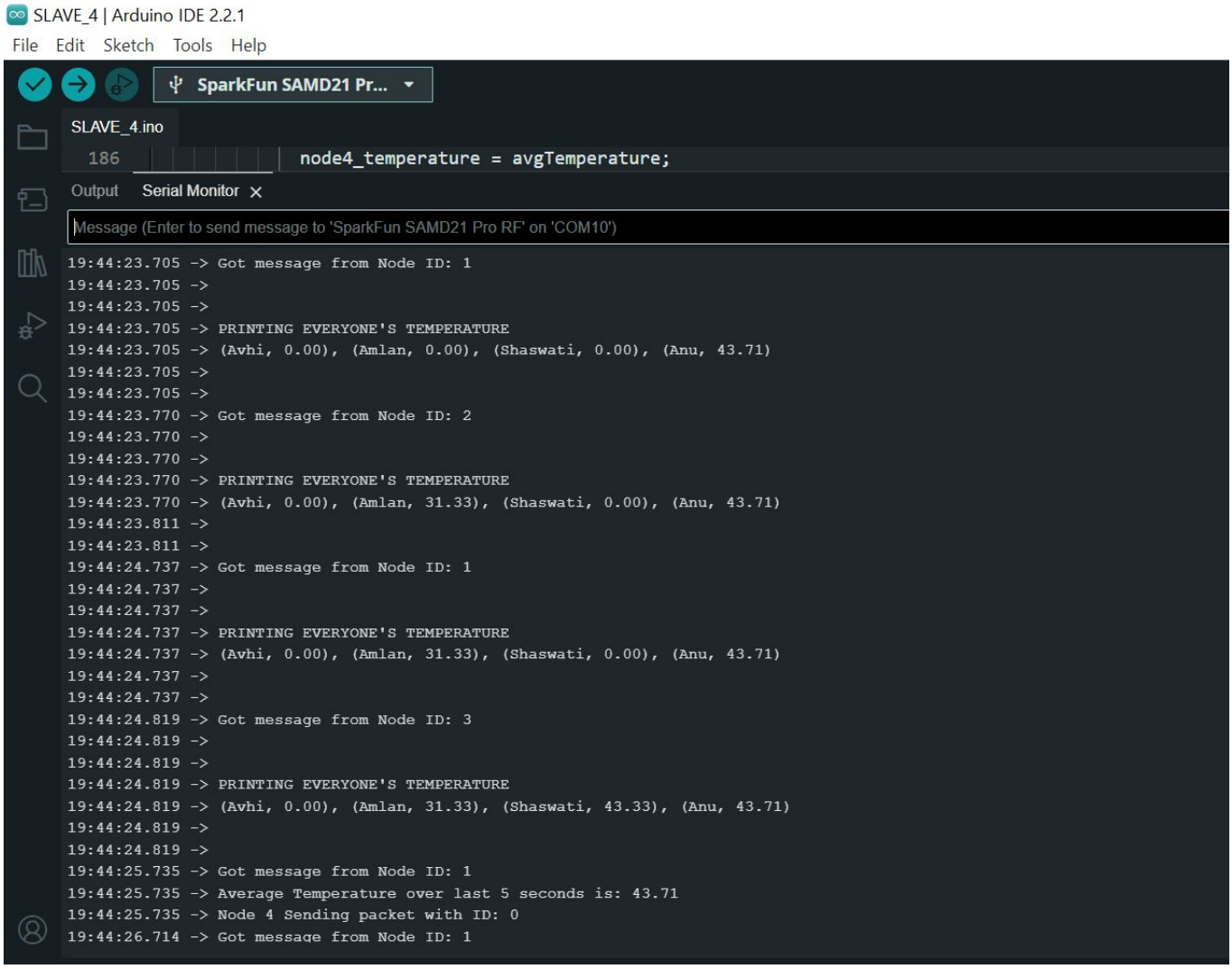

Fig 7. Child Node 4, showing authentication and packet transmission request from Master/Node1. Also subsequent logs show the temperature from other Nodes in required format.

Task 3 : Error Logging

Requirements

1. System reset
    1. Implement a WDT and enable the WDT interrupt as we learnt in lab 1 and lab 2
2. Communication error
    1. Packet reception failure: see example code
    2. Missing packet
3. Error log structure
    1. Timestamp

    2. Error code
4. Store only the error code in the flash storage

Development Plan:

a. Procedure of Solving the Problem

1. WDT is implemented for every 2s and is kicked in `TC4_handler()`
2. When WDT interrupt occurs, `DSU_STATUSA` and `DSU_STATUSB` register data are collected.
3. Error codes from `RCAUSE_REG, REG_DSU_STATUSA and REG_DSU_STATUS` is added to the transmission packet from every slave node to the master node. The master node will save those error codes in the flash storage.
4. Communication error logging code is like example code, but packet IDs (current and previous id) are tracked for each transmission and reception.
5. In the serial monitor status of the flash storage are printed with the time stamp.

b. Configuration Table

Bit Number and Associated Error Information

Bit Number	Concerned Register	Error Information
0	RCAUSE	System Reset
1	RCAUSE	External Reset
2	RCAUSE	Watchdog Reset
3	STATUS A	This bit is set when a command that is not allowed in protected state is issued.
4	STATUS A	This bit is set when a DSU operation failure is detected.
5	STATUS A	This bit is set when a bus error is detected.
6	STATUS A	This bit is set when a debug adapter Cold-Plugging is detected, which extends the CPU reset phase.
7	STATUS B	This value is set when Hot-Plugging is enabled.
8	STATUS B	This bit is set when DCC1 is written. This bit is cleared when DCC1 is read.
9	STATUS B	This bit is set when DCC0 is written. This bit is cleared when DCC0 is read.
10	STATUS B	This bit is set when a debugger probe is detected.
11	STATUS B	Missing Packet Error
12	STATUS B	Packet Receive Error

b. Run-time Errors

2. Development Process:

1. For task 3, we used code from LAB1 for setting up the WDT but to set up the early warning interrupt we are using WDT_Handler that we got from the Arduino Zero website.
2. For transmitting error codes, we used a 16 cells char array with each cell used store a flag for a particular error.
3. Then, this char array was passed on to master node where we traverse this error code using if-else constraints to display the right error message. The following table shows the cell number and the respective error causes.

3. Test Plan:

1. Test of packets: The packets were printed to serial monitor in all the nodes to check whether all the error logging messages were contained on the packet.
2. Test our code with system reset, WDT reset and external reset.

3. Print all the packets gathered by leader node to check if all the communications are happening correctly
4. Write all the error codes to flash storage

Component Test Results and Comments

Component	Test	Results	Comments
communication	Missing packet	Pass	No errors due to good collision avoidance in code
	Packet reception failure	Pass	No errors due to good collision avoidance in code
WDT	Resetting	-	Could not test
STATUSA	-	-	Could not test
STATUSB	-	-	Could not test

Code solutions:

1. **System reset** : The below code checks the reason of reset from the `RCAUSE` register. The `WDT_Handler` checks the `REG_DSU_STATUSA` and the `REG_DSU_STATUSB` . The Watchdog is there to track the failure from the nodes, below code is from one of the codes.

Code

```
void reset_reason(){
//determine last reset type
uint8_t reset_reg = PM->RCAUSE.reg;
SerialUSB.println(bitRead(reset_reg, 6));
if(bitRead(reset_reg, 6)) {error_bits[0]=1;}//---------------------------------------------changed
if(bitRead(reset_reg, 5)) {error_bits[1]=1;}
if(bitRead(reset_reg, 4)) {error_bits[2]=1;}
//  bitWrite(error_bits, 0, bitRead(reset_reg, 6));
//  bitWrite(error_bits, 1, bitRead(reset_reg, 5));
//  bitWrite(error_bits, 2, bitRead(reset_reg, 4));
//   bitWrite(error_bits, 13, 1);
}

void WDT_Handler() {
  //SerialUSB.println("WDT Interrupt");
  uint8_t reg_statusA = REG_DSU_STATUSA;
  uint8_t reg_statusB = REG_DSU_STATUSB;
  uint16_t status_errors = 0x0000;
  status_errors = reg_statusA;
  status_errors = status_errors << 8;
  status_errors |= reg_statusB;

  if(bitRead(status_errors, 12)) {error_bits[3]=1;}
  if(bitRead(status_errors, 11)) {error_bits[4]=1;}
  if(bitRead(status_errors, 10)) {error_bits[5]=1;}
  if(bitRead(status_errors, 9)) {error_bits[6]=1;}
  if(bitRead(status_errors, 4)) {error_bits[7]=1;}
  if(bitRead(status_errors, 3)) {error_bits[8]=1;}
  if(bitRead(status_errors, 2)) {error_bits[9]=1;}
  if(bitRead(status_errors, 1)) {error_bits[10]=1;}
}
```

2.**Communication Error**

Code from Master Node when recieveing from each node

```
            if(current_packet_id-previouse_packet_id > 1) {
              receivedPacket->error[11]='1';
            }
           // check the missing packet
           // write_error(receivedPacket->nodeID, 19) ;
             previouse_packet_id = current_packet_id;
           }
          else
          {
          SerialUSB.println("Recieve failed");
          char temp1[]={'0','0','0','0','0','0','0','0','0','0','0','0','1','0','0','0'};
          write_error(5, temp1);                              // Save the error code as message receive failed
          }
```

3. **Error log structure** - Error Log structure from Master Node which checks the error code from each node with the node error database to log the exact error in the flash memory later and print it.

```
void print_errors(char* err){
  SerialUSB.print("Timestamp :")
  SerialUSB.print(millis())
  SerialUSB.println("Error =  ");
  // SerialUSB.println(err);
  // SerialUSB.println(err.charAt(1));

 if (err[0] == '1') {
 SerialUSB.println("System Reset");
 }

 if (err[1] == '1') {
 SerialUSB.println("Watchdog Reset");
 }
 if (err[2] == '1') {
 SerialUSB.println("External Reset");
 }
 if (err[3] == '1') {
 SerialUSB.println("Command not allowed error: PROTECTED STATE");
 }
 if (err[4] == '1') {
 SerialUSB.println("DSU Operation Failure Error");
 }
 if (err[5] == '1') {
 SerialUSB.println("BUS ERROR Detected!");
 }
 if (err[6] == '1') {
 SerialUSB.println("COLD PLUGGING ERROR!!");
 }
 if (err[7] == '1') {
 SerialUSB.println("HOT PLUGGING ERROR!");
 }
 if (err[8] == '1') {
 SerialUSB.println("DCC1 Written Error");
 }
 if (err[9] == '1') {
 SerialUSB.println("DCC0 Written Error");
 }
 if (err[10] == '1') {
 SerialUSB.println("Debugger Probe Error");
 }
 if (err[11] == '1') {
 SerialUSB.println("Missing Packet Error!");
 }
 if (err[12] == '1') {
 SerialUSB.println("Packet Receival Error");
 }
}
```

4. **Store only the error code in the flash storage**

```
void write_error(uint8_t Node_name, char* ecodeX) //---------changed
{
  int ecode=atoi(ecodeX);
  if(Node_name==3)
  {error_Amlan.write(ecode);}

   if(Node_name==2)
  {error_Shaswati.write(ecode);}

    if(Node_name==4)
  {error_Anuruddha.write(ecode);}

    if(Node_name==1)
  {error_Avhishek.write(ecode);}

    if(Node_name==5)
  {error_receive.write(ecode);}
}

FlashStorage(error_Amlan, int);                // Reserve a portion of flash memory, remove/comment when upload to the transmitter
FlashStorage(error_Shaswati, int);
FlashStorage(error_Anuruddha, int);
FlashStorage(error_Avhishek, int);
FlashStorage(error_receive, int);

SerialUSB.println();
//FLASH STORAGE DETAILS PRINT
SerialUSB.println("FLASH STORAGE RESULTS");
SerialUSB.print("AMLAN: ");SerialUSB.print(error_Amlan.read());
SerialUSB.print(", SHASWATI: ");SerialUSB.print(error_Shaswati.read());
SerialUSB.print(", ANURUDDHA: ");SerialUSB.print(error_Anuruddha.read());
SerialUSB.print(", AVHISHEK: ");SerialUSB.print(error_Avhishek.read());
SerialUSB.print(", ERROR RECEIVE ");SerialUSB.println(error_receive.read());
```

Task 4 : Timer

Requirements

1. For periodical tasks, use the timer technique we learnt from last lab.

Development Plan:

a. Procedure of Solving the Problem

1. The function `startTimer(int frequencyHz)` is the entry point for setting up the timer. It calls two other functions: `configureClock()` and `configureTimer(frequencyHz)`.
2. The function `configureClock()` sets up the general clock by writing to the `REG_GCLK_CLKCTRL` register and waiting for synchronization.
3. The function `configureTimer(int frequencyHz)` sets up the timer by configuring its control register, setting its frequency, and enabling interrupts.
4. Inside `configureTimer()`, the function `setTimerFrequency(TC, frequencyHz)`` is called to set the timer frequency based on the input frequency in Hz.
5. The code enables timer interrupts by setting the `TC->INTENSET.bit.MC0 = 1` and specifies that the interrupt to be used is `TC3_IRQn`.

6. The function `TC3_Handler()` serves as the interrupt handler. It toggles an LED and prints a message to the serial port.
7. The function `setTimerFrequency(TcCount16* TC, int frequencyHz)` calculates the compare value for the timer based on the desired frequency.

b. Configuration Table

Requirement	Register Name	Register Function	Register Value
Clock Configuration	`REG_GCLK_CLKCTRL`	Configure Generic Clock Control	`GCLK_CLKCTRL_CLKEN | GCLK_CLKCTRL_GEN_GCLK0 | GCLK_CLKCTRL_ID_TCC2_TC3`
Timer Configuration	`TC->CTRLA.reg`	Timer Control A Register	`TC_CTRLA_MODE_COUNT16 | TC_CTRLA_WAVEGEN_MFRQ | TC_CTRLA_PRESCALER_DIV1024`
Timer Frequency	`TC->CC[0].reg`	Timer Compare/Capture Register	`compareValue` (Calculated based on `frequencyHz`, `CPU_HZ`, and `TIMER_PRESCALER_DIV`)
Timer Interrupt	`TC->INTENSET.reg`	Timer Interrupt Enable Set Register	`TC_INTENSET.bit.MC0 = 1`
NVIC Configuration	`NVIC_EnableIRQ()`	Nested Vector Interrupt Controller	`TC3_IRQn`
Clock Sync	`GCLK->STATUS.bit.SYNCBUSY`	Clock Synchronization Busy Status	`1` (Wait until it becomes 0)
Timer Sync	`TC->STATUS.bit.SYNCBUSY`	Timer Synchronization Busy Status	`1` (Wait until it becomes 0)

b. Run-time Errors

Error Code	Error Description
`clock_sync`	Synchronization busy flag for the clock never clears, causing an infinite loop in `configureClock()`
`timer_sync`	Synchronization busy flag for the timer never clears, causing an infinite loop in `configureTimer()` and `setTimerFrequency()`
`clock_config`	Incorrect configuration of the general clock in `configureClock()` leading to undesired behavior
`timer_config`	Incorrect timer configuration in `configureTimer()` leading to undesired behavior
`freq_calc`	Incorrect calculation of `compareValue` in `setTimerFrequency()` leading to incorrect timer frequency
`irq_fail`	Failure to enable the interrupt correctly, causing `TC3_Handler()` not to be called
`flag_mismatch`	`TC->INTFLAG.bit.MC0` is not set or cleared correctly, causing issues in `TC3_Handler()`

2. Development Process:

Subtask 1:

```
void startTimer(int frequencyHz) {
  configureClockTC4();
  configureTimerTC4(frequencyHz);
}
void configureClockTC4() {
  REG_GCLK_CLKCTRL = (uint16_t)(GCLK_CLKCTRL_CLKEN | GCLK_CLKCTRL_GEN_GCLK0 | GCLK_CLKCTRL_ID_TC4_TC5);
  while (GCLK->STATUS.bit.SYNCBUSY == 1);
}

void configureTimerTC4(int frequencyHz) {
  TcCount16* TC = (TcCount16*)TC4;
  TC->CTRLA.reg &= ~TC_CTRLA_ENABLE;
  while (TC->STATUS.bit.SYNCBUSY == 1);

  TC->CTRLA.reg |= TC_CTRLA_MODE_COUNT16 | TC_CTRLA_WAVEGEN_MFRQ | TC_CTRLA_PRESCALER_DIV1024;
  while (TC->STATUS.bit.SYNCBUSY == 1);

  setTimerFrequencyTC4(TC, frequencyHz);
  TC->INTENSET.reg = 0;
  TC->INTENSET.bit.MC0 = 1;

  NVIC_EnableIRQ(TC4_IRQn);
  TC->CTRLA.reg |= TC_CTRLA_ENABLE;
  while (TC->STATUS.bit.SYNCBUSY == 1);
}

void setTimerFrequencyTC4(TcCount16* TC, int frequencyHz) {
  int compareValue = (CPU_HZ / (TIMER_PRESCALER_DIV * frequencyHz)) - 1;
  TC->CC[0].reg = compareValue;
  while (TC->STATUS.bit.SYNCBUSY == 1);
}

void TC4_Handler() {
  static bool isLEDOn = false;
  TcCount16* TC = (TcCount16*)TC4;
  if (TC->INTFLAG.bit.MC0 == 1) {
    TC->INTFLAG.bit.MC0 = 1;
    digitalWrite(PIN_LED_13, isLEDOn);
    SerialUSB.println(isLEDOn ? "Blue LED is on" : "Blue LED is off");
    isLEDOn = !isLEDOn;
    canReadTemp = true;
  }
}
```

3. Test Plan:

1. **LED Blinking using Timer**:
   - Verify that the Yellow LED toggles on and off at an interval of 1000 milliseconds.
2. **Serial Output**:
   - Verify that the Serial Monitor displays the correct LED status messages ("Blue LED is on/off") in real-time.

Component	Test Description	Result	Comment
Blue LED	Toggle on/off at 1s interval	Pass	LED toggled as expected at 1-second intervals
Serial Output	Display "Blue LED is on/off"	Pass	Messages displayed correctly in the serial monitor
System Level	LEDs toggle at correct intervals of 1sec	Pass	Both LEDs toggled at their respective intervals without conflict

Screenshot

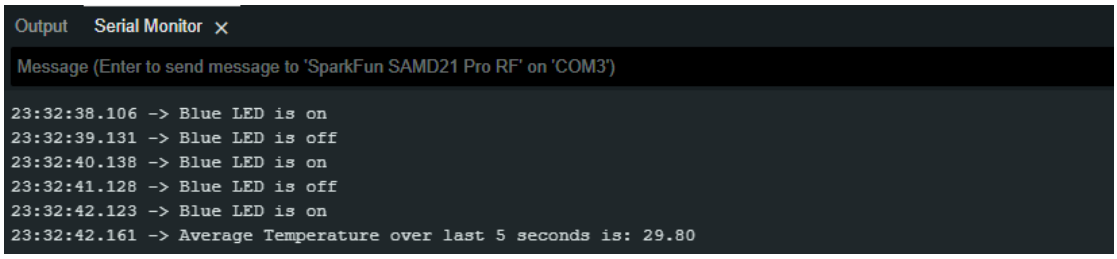

Fig.8 - Blue LED turning on and off every 1 sec based on ticks from the timer.

Appendix

Task 1 and Task 4 :

```
# define CPU_HZ 48000000
<p class="crossnote-header " id="define-cpu_hz-48000000"></p>

# define TIMER_PRESCALER_DIV 1024
<p class="crossnote-header " id="define-timer_prescaler_div-1024"></p>

# define WINDOW_SIZE 5  // Size of the sliding window
<p class="crossnote-header " id="define-window_size-5---size-of-the-sliding-window"></p>


# include <TemperatureZero.h>
<p class="crossnote-header " id="include-temperaturezeroh"></p>


TemperatureZero TempZero = TemperatureZero();
volatile bool canReadTemp = false;
float tempBuffer[WINDOW_SIZE];  // Circular buffer to hold temperature values
int tempIndex = 0;  // Index to keep track of the oldest temperature value
float tempSum = 0;  // Sum of temperatures in the buffer
int tempCount = 0;  // Count of temperatures added to the buffer
int reportCount = 0;  // Count of readings before reporting
float avgTemperature = 0;  // To hold the average temperature
bool newAvgAvailable = false;  // Flag to indicate a new average is available


void setup() {
  SerialUSB.begin(9600);
  TempZero.init();
  pinMode(PIN_LED_13, OUTPUT);
  startTimer(1);
}

void loop() {
  Readtemp();
  if (newAvgAvailable) {
    SerialUSB.print("Average Temperature over last 5 seconds is: ");
    SerialUSB.println(avgTemperature);
    newAvgAvailable = false;  // Reset the flag
  }
}

void startTimer(int frequencyHz) {
  configureClock();
  configureTimer(frequencyHz);
}

void configureClock() {
  REG_GCLK_CLKCTRL = (uint16_t)(GCLK_CLKCTRL_CLKEN | GCLK_CLKCTRL_GEN_GCLK0 | GCLK_CLKCTRL_ID_TCC2_TC3);
  while (GCLK->STATUS.bit.SYNCBUSY == 1);
}

void configureTimer(int frequencyHz) {
  TcCount16* TC = (TcCount16*)TC3;
  TC->CTRLA.reg &= ~TC_CTRLA_ENABLE;
  while (TC->STATUS.bit.SYNCBUSY == 1);

  TC->CTRLA.reg |= TC_CTRLA_MODE_COUNT16 | TC_CTRLA_WAVEGEN_MFRQ | TC_CTRLA_PRESCALER_DIV1024;
  while (TC->STATUS.bit.SYNCBUSY == 1);

  setTimerFrequency(TC, frequencyHz);
  TC->INTENSET.reg = 0;
  TC->INTENSET.bit.MC0 = 1;

  NVIC_EnableIRQ(TC3_IRQn);
  TC->CTRLA.reg |= TC_CTRLA_ENABLE;
  while (TC->STATUS.bit.SYNCBUSY == 1);
}

void setTimerFrequency(TcCount16* TC, int frequencyHz) {
  int compareValue = (CPU_HZ / (TIMER_PRESCALER_DIV * frequencyHz)) - 1;
```

```
  TC->CC[0].reg = compareValue;
  while (TC->STATUS.bit.SYNCBUSY == 1);
}

void TC3_Handler() {
  static bool isLEDOn = false;
  TcCount16* TC = (TcCount16*)TC3;
  if (TC->INTFLAG.bit.MC0 == 1)
  {
    TC->INTFLAG.bit.MC0 = 1;
    digitalWrite(PIN_LED_13, isLEDOn);
    SerialUSB.println(isLEDOn ? "Blue LED is on" : "Blue LED is off");
    isLEDOn = !isLEDOn;
    canReadTemp = true;
  }
}

float Readtemp() {
  if (canReadTemp) {
    float temperature = TempZero.readInternalTemperature();
    SerialUSB.print("Internal Temperature is: ");
    SerialUSB.println(temperature);
    // Remove the oldest temperature from the sum
    tempSum -= tempBuffer[tempIndex];

    // Add the new temperature to the buffer and sum
    tempBuffer[tempIndex] = temperature;
    tempSum += temperature;

    // Update the index for the oldest temperature
    tempIndex = (tempIndex + 1) % WINDOW_SIZE;

    // Update the count of temperatures added to the buffer
    if (tempCount < WINDOW_SIZE) {
      tempCount++;
    }

    // Increment the report counter
    reportCount++;

    // If 5 readings have been taken, calculate the average temperature
    if (reportCount >= WINDOW_SIZE) {
      avgTemperature = tempSum / tempCount;
      newAvgAvailable = true;  // Set the flag
      reportCount = 0;  // Reset the report counter
    }

    canReadTemp = false;
  }
  return avgTemperature;
}
```

Task 2 and 3

Master Node:

```
//MASTER  NODE = 1

# define CPU_HZ 48000000
<p class="crossnote-header " id="define-cpu_hz-48000000-1"></p>

# define TIMER_PRESCALER_DIV 1024
<p class="crossnote-header " id="define-timer_prescaler_div-1024-1"></p>

# define WINDOW_SIZE 5  // Size of the sliding window
<p class="crossnote-header " id="define-window_size-5---size-of-the-sliding-window-1"></p>

# include <TemperatureZero.h>                       //Arduino library for internal temperature measurment of the family SAMD21 and SAMD51
<p class="crossnote-header " id="include-temperaturezeroh----------arduino-library-for-internal-temperature-measurment-of-the-famil

TemperatureZero TempZero = TemperatureZero();

# include <SPI.h>                                   //This library allows you to communicate with SPI devices
<p class="crossnote-header " id="include-spih----------this-library-allows-you-to-communicate-with-spi-devices"></p>

# include <RH_RF95.h>
<p class="crossnote-header " id="include-rh_rf95h"></p>

# include <FlashStorage.h>
<p class="crossnote-header " id="include-flashstorageh"></p>



FlashStorage(error_Amlan, int);                 // Reserve a portion of flash memory, remove/comment when upload to the transmitter
FlashStorage(error_Shaswati, int);
FlashStorage(error_Anuruddha, int);
FlashStorage(error_Avhishek, int);
FlashStorage(error_receive, int);

struct Packet {
  uint8_t nodeID;
  uint16_t packetID;
  uint32_t timestamp;
  float payload;
  char error[16];
  uint8_t authID;

  // Constructor to initialize the values
  Packet() : nodeID(0), packetID(0), timestamp(0), payload(0.0), authID(0) {
    // Initialize the error array to zeros
    memset(error, '0', sizeof(error));
  }
};

volatile bool canReadTemp = false;

volatile bool Slave2 = false;
volatile bool Slave3 = false;
volatile bool Slave4 = false;
volatile bool Master = false;

float node1_temperature = 0;
float node2_temperature = 0;
float node3_temperature = 0;
float node4_temperature = 0;


float tempBuffer[WINDOW_SIZE];  // Circular buffer to hold temperature values
int tempIndex = 0;  // Index to keep track of the oldest temperature value
float tempSum = 0;  // Sum of temperatures in the buffer
int tempCount = 0;  // Count of temperatures added to the buffer
int reportCount = 0;  // Count of readings before reporting
float avgTemperature = 0;  // To hold the average temperature
bool newAvgAvailable = false;  // Flag to indicate a new average is available

//Define functions used in the program
void startTimer(int frequencyHz);
```

```
void configureClockTC4();
void configureTimerTC4(int frequencyHz);                //
void setTimerFrequencyTC4(TcCount16* TC, int frequencyHz);
void TC4_Handler();
float Readtemp();
void write_error(int Node_name, int ecode);
void print_errors(char* err);

int previouse_packet_id = 0;
int current_packet_id = 0;

// We need to provide the RFM95 module's chip select and interrupt pins to the
// rf95 instance below.On the SparkFun ProRF those pins are 12 and 6 respectively.
RH_RF95 rf95(12, 6);
int LED = 13; //Status LED is on pin 13
long timeSinceLastPacket = 0; //Tracks the time stamp of last packet received

float frequency = 910; //Broadcast frequency
uint16_t packetCounter = 0;  // Initialize packet counter

// long timeSinceLastPacket = 0;

void setup() {
     GCLK->GENDIV.reg = GCLK_GENDIV_ID(2) | GCLK_GENDIV_DIV(4);
     GCLK->GENCTRL.reg = GCLK_GENCTRL_ID(2) |
     GCLK_GENCTRL_GENEN |
     GCLK_GENCTRL_SRC_OSCULP32K |
     GCLK_GENCTRL_DIVSEL;
     while(GCLK->STATUS.bit.SYNCBUSY);                              // wait while synchronizing the GCLK->STATUS.bit
                                                                    // WDT clock = clock gen 2
     GCLK->CLKCTRL.reg = GCLK_CLKCTRL_ID_WDT |
     GCLK_CLKCTRL_CLKEN |
     GCLK_CLKCTRL_GEN_GCLK2;
     WDT->CTRL.reg = 0;
     WDT->CONFIG.bit.PER = 0x8; // Set period for chip reset from the datasheet
     WDT->INTENSET.bit.EW = 1; // Enable early warning interrupt
     WDT->CTRL.bit.WEN = 0;
     SerialUSB.begin(9600);
     TempZero.init();
      // It may be difficult to read serial messages on startup. The following line
    // will wait for serial to be ready before continuing. Comment out if not needed.
    // while (!SerialUSB);
     SerialUSB.println("RFM Client!");
    //Initialize the Radio.
     if (rf95.init() == false) {
        SerialUSB.println("Radio Init Failed - Freezing");
        while (1);
      }
     else {
        //An LED inidicator to let us know radio initialization has completed.
        // rf95.setModemConfig(Bw125Cr48Sf4096); // slow and reliable?
        SerialUSB.println("Transmitter up!");
        digitalWrite(LED, HIGH);
        delay(500);
        digitalWrite(LED, LOW);
        delay(500);
        // Set frequency
     rf95.setFrequency(frequency);
        // Transmitter power can range from 14-20dbm.
     rf95.setTxPower(20, false);
     }
     pinMode(PIN_LED_13, OUTPUT);
     WDT->CTRL.bit.ENABLE = 1;
     startTimer(1);
}

void loop()
{ avgTemperature = Readtemp();
  // for Node =  slave 2
  if(reportCount == 1 && Slave2 == false)
  {    Master = false;
       Slave2 = true;
      // Send the authentication packet
      SerialUSB.println("Asking Node 2 to send it's temp. ");

      // Create the packet
```

```
  Packet packet;
  packet.nodeID = 1;  // Node 1
  packet.packetID = packetCounter++;
  packet.timestamp = millis();  // Current time in milliseconds
  packet.payload = avgTemperature;
  //packet.error = 0;
  packet.authID = 2;

  // Serialize the packet into a byte array
  uint8_t toSend[sizeof(Packet)];
  memcpy(toSend, &packet, sizeof(Packet));

  // Print and send the message
  SerialUSB.print("Sending authentication packet to Node 2: Packet Number :");
  SerialUSB.println(packet.packetID);
  rf95.send(toSend, sizeof(Packet));

  // Recieve the data packet from Node 2
  // SerialUSB.println(rf95.available());
  while(rf95.available() == 0){}

// Recieve the data packet from Node 2
  if (rf95.available()){
    // Should be a message for us now
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
    if (rf95.recv(buf, &len))
    {
        // Cast the received buffer to the Packet struct type
        Packet *receivedPacket = (Packet *)buf;
        buf[len] = '\0';  // Null-terminate the received string
        digitalWrite(LED, HIGH); //Turn on status LED
        timeSinceLastPacket = millis(); //Timestamp this packet

        //SAVE THE TEMPERATURE OF NODE 2
        node2_temperature = receivedPacket->payload;
        node1_temperature = avgTemperature;

        // Print the received packet details
        SerialUSB.print("Got message from Node ID: ");
        SerialUSB.print(receivedPacket->nodeID);
        SerialUSB.print(", Packet ID: ");
        SerialUSB.print(receivedPacket->packetID);
        SerialUSB.print(", Timestamp: ");
        SerialUSB.print(receivedPacket->timestamp);
        SerialUSB.print(", Payload (Temperature): ");
        SerialUSB.print(receivedPacket->payload);

        SerialUSB.println();
        print_errors(receivedPacket->error);
        //FLASH STORAGE DETAILS PRINT
        SerialUSB.println("FLASH STORAGE RESULTS");
        SerialUSB.print("AMLAN: ");SerialUSB.print(error_Amlan.read());
        SerialUSB.print(", SHASWATI: ");SerialUSB.print(error_Shaswati.read());
        SerialUSB.print(", ANURUDDHA: ");SerialUSB.print(error_Anuruddha.read());
        SerialUSB.print(", AVHISHEK: ");SerialUSB.print(error_Avhishek.read());
        SerialUSB.print(", ERROR RECEIVE ");SerialUSB.println(error_receive.read());

        SerialUSB.println();
        SerialUSB.println();

        SerialUSB.println("PRINTING EVERYONE'S TEMPERATURE");
        SerialUSB.print("(Avhi, "); SerialUSB.print(node1_temperature, 2);
        SerialUSB.print("), (Amlan, "); SerialUSB.print(node2_temperature, 2);
        SerialUSB.print("), (Shaswati, "); SerialUSB.print(node3_temperature, 2);
        SerialUSB.print("), (Anu, "); SerialUSB.print(node4_temperature, 2);
        SerialUSB.println(")");

        SerialUSB.println();
        SerialUSB.println();


        write_error(receivedPacket->nodeID, receivedPacket->error) ;                    //Write the received packet error in the correct

           current_packet_id = receivedPacket->packetID;                //Assign the current packet ID to the current_packet_id variable
```

```
                if(current_packet_id-previouse_packet_id > 1) {
                  receivedPacket->error[11]='1';
                }           //check the missing packet
               //  write_error(receivedPacket->nodeID, 19) ;
                previouse_packet_id = current_packet_id;
       }
       else
       {
       SerialUSB.println("Recieve failed");
       char temp1[]={'0','0','0','0','0','0','0','0','0','0','0','0','1','0','0','0'};
       write_error(5, temp1);                              // Save the error code as message receive failed
       }
   }
   timeSinceLastPacket = millis();
}

// for Node = slave 3
else if(reportCount == 2 && Slave3 == false)
{   Slave3 = true;
    // Send the authentication packet
    SerialUSB.println("Asking Node 3 to send it's temp. ");

    // Create the packet
    Packet packet;
    packet.nodeID = 1;  // Node 1
    packet.packetID = packetCounter++;
    packet.timestamp = millis();  // Current time in milliseconds
    packet.payload = avgTemperature;
    // packet.error = 0;
    packet.authID = 3;

    // Serialize the packet into a byte array
    uint8_t toSend[sizeof(Packet)];
    memcpy(toSend, &packet, sizeof(Packet));

    // Print and send the message
    SerialUSB.print("Sending authentication packet to Node 3 : Packet Number : ");
    SerialUSB.println(packet.packetID);
    rf95.send(toSend, sizeof(Packet));

    // Recieve the data packet from Node 3
    // SerialUSB.println(rf95.available());
    while(rf95.available() == 0){
    }

  // Recieve the data packet from Node 3
    if (rf95.available()){
      // Should be a message for us now
      uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
      uint8_t len = sizeof(buf);
      if (rf95.recv(buf, &len))
      {
          // Cast the received buffer to the Packet struct type
          Packet *receivedPacket = (Packet *)buf;
          buf[len] = '\0';  // Null-terminate the received string
          digitalWrite(LED, HIGH); //Turn on status LED
          timeSinceLastPacket = millis(); //Timestamp this packet

          //SAVE THE TEMPERATURE OF NODE 2
          node3_temperature = receivedPacket->payload;
          node1_temperature = avgTemperature;

          // Print the received packet details
          SerialUSB.print("Got message from Node ID: ");
          SerialUSB.print(receivedPacket->nodeID);
          SerialUSB.print(", Packet ID: ");
          SerialUSB.print(receivedPacket->packetID);
          SerialUSB.print(", Timestamp: ");
          SerialUSB.print(receivedPacket->timestamp);
          SerialUSB.print(", Payload (Temperature): ");
          SerialUSB.print(receivedPacket->payload);
      //     SerialUSB.print(" RSSI: ");
      //     SerialUSB.print(rf95.lastRssi(), DEC);
          SerialUSB.println();
           print_errors(receivedPacket->error);
          //FLASH STORAGE DETAILS PRINT
```

```
          SerialUSB.println("FLASH STORAGE RESULTS");
          SerialUSB.print("AMLAN: ");SerialUSB.print(error_Amlan.read());
          SerialUSB.print(", SHASWATI: ");SerialUSB.print(error_Shaswati.read());
          SerialUSB.print(", ANURUDDHA: ");SerialUSB.print(error_Anuruddha.read());
          SerialUSB.print(", AVHISHEK: ");SerialUSB.print(error_Avhishek.read());
          SerialUSB.print(", ERROR RECEIVE ");SerialUSB.println(error_receive.read());


          SerialUSB.println();
          SerialUSB.println();

          SerialUSB.println("PRINTING EVERYONE'S TEMPERATURE");
          SerialUSB.print("(Avhi, "); SerialUSB.print(node1_temperature, 2);
          SerialUSB.print("), (Amlan, "); SerialUSB.print(node2_temperature, 2);
          SerialUSB.print("), (Shaswati, "); SerialUSB.print(node3_temperature, 2);
          SerialUSB.print("), (Anu, "); SerialUSB.print(node4_temperature, 2);
          SerialUSB.println(")");

          SerialUSB.println();
          SerialUSB.println();

          write_error(receivedPacket->nodeID, receivedPacket->error) ;                      //Write the received packet error in the correct

          current_packet_id = receivedPacket->packetID;                //Assign the current packet ID to the current_packet_id variable

          if(current_packet_id-previouse_packet_id > 1) {
            receivedPacket->error[11]='1';
          }
         // check the missing packet
         // write_error(receivedPacket->nodeID, 19) ;
           previouse_packet_id = current_packet_id;
         }
        else
        {
        SerialUSB.println("Recieve failed");
        char temp1[]={'0','0','0','0','0','0','0','0','0','0','0','0','1','0','0','0'};
        write_error(5, temp1);                              // Save the error code as message receive failed
        }
    }
    timeSinceLastPacket = millis();
}

// for Node = slave 4
else if(reportCount == 3 && Slave4 == false)
  {   Slave4 = true; Master = false;
     // Send the authentication packet
     SerialUSB.println("Asking Node 4 to send it's temp. ");

     // Create the packet
     Packet packet;
     packet.nodeID = 1;  // Node 1
     packet.packetID = packetCounter++;
     packet.timestamp = millis();  // Current time in milliseconds
     packet.payload = avgTemperature;
     // packet.error = 0;
     packet.authID = 4;

     // Serialize the packet into a byte array
     uint8_t toSend[sizeof(Packet)];
     memcpy(toSend, &packet, sizeof(Packet));

     // Print and send the message
     SerialUSB.print("Sending authentication packet to Node 4 : Packet Number : ");
     SerialUSB.println(packet.packetID);
     rf95.send(toSend, sizeof(Packet));


     // Recieve the data packet from Node 4
     // SerialUSB.println(rf95.available());
     while(rf95.available() == 0){}
     // SerialUSB.println(rf95.available());

     if (rf95.available()){
       // Should be a message for us now
       uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
       uint8_t len = sizeof(buf);
```

```
      if (rf95.recv(buf, &len))
      {
          // Cast the received buffer to the Packet struct type
          Packet *receivedPacket = (Packet *)buf;
          buf[len] = '\0';  // Null-terminate the received string
          digitalWrite(LED, HIGH); //Turn on status LED
          timeSinceLastPacket = millis(); //Timestamp this packet

          //SAVE THE TEMPERATURE OF NODE 2
          node4_temperature = receivedPacket->payload;
          node1_temperature = avgTemperature;

          // Print the received packet details
          SerialUSB.print("Got message from Node ID: ");
          SerialUSB.print(receivedPacket->nodeID);
          SerialUSB.print(", Packet ID: ");
          SerialUSB.print(receivedPacket->packetID);
          SerialUSB.print(", Timestamp: ");
          SerialUSB.print(receivedPacket->timestamp);
          SerialUSB.print(", Payload (Temperature): ");
          SerialUSB.print(receivedPacket->payload);
      //     SerialUSB.print(" RSSI: ");
      //     SerialUSB.print(rf95.lastRssi(), DEC);
          SerialUSB.println();
           print_errors(receivedPacket->error);
          //FLASH STORAGE DETAILS PRINT
          SerialUSB.println("FLASH STORAGE RESULTS");
          SerialUSB.print("AMLAN: ");SerialUSB.print(error_Amlan.read());
          SerialUSB.print(", SHASWATI: ");SerialUSB.print(error_Shaswati.read());
          SerialUSB.print(", ANURUDDHA: ");SerialUSB.print(error_Anuruddha.read());
          SerialUSB.print(", AVHISHEK: ");SerialUSB.print(error_Avhishek.read());
          SerialUSB.print(", ERROR RECEIVE ");SerialUSB.println(error_receive.read());


          SerialUSB.println();
          SerialUSB.println();

          SerialUSB.println("PRINTING EVERYONE'S TEMPERATURE");
          SerialUSB.print("(Avhi, "); SerialUSB.print(node1_temperature, 2);
          SerialUSB.print("), (Amlan, "); SerialUSB.print(node2_temperature, 2);
          SerialUSB.print("), (Shaswati, "); SerialUSB.print(node3_temperature, 2);
          SerialUSB.print("), (Anu, "); SerialUSB.print(node4_temperature, 2);
          SerialUSB.println(")");

          SerialUSB.println();
          SerialUSB.println();


          write_error(receivedPacket->nodeID, receivedPacket->error) ;                      //Write the received packet error in the correct

             current_packet_id = receivedPacket->packetID;                //Assign the current packet ID to the current_packet_id variable

                   if(current_packet_id-previouse_packet_id > 1) {
                     receivedPacket->error[11]='1';
                   }           //check the missing packet
                  //  write_error(receivedPacket->nodeID, 19) ;
                   previouse_packet_id = current_packet_id;
       }
       else
       {
       SerialUSB.println("Recieve failed");
       char temp1[]={'0','0','0','0','0','0','0','0','0','0','0','0','1','0','0','0'};
       write_error(5, temp1);                              // Save the error code as message receive failed
       }
   }
   timeSinceLastPacket = millis();
}

// Master sends it average temperature
else if(reportCount == 4 && Master == false)
{  // set the slaves false for the next 5 secs
    Slave2 = false;
    Slave3 = false;
    // Slave4 = false;
    Master = true;
    // Send the authentication packet
```

```
      SerialUSB.println("Sending Master's Average Temperature. ");

      //SAVE THE TEMPERATURE OF NODE 2
      node1_temperature = avgTemperature;

      // Create the packet
      Packet packet;
      packet.nodeID = 1;  // Node 1
      packet.packetID = packetCounter++;
      packet.timestamp = millis();  // Current time in milliseconds
      packet.payload = avgTemperature;
      // packet.error = 0;
      packet.authID = 10;  // master is sending it's temperature

      // Serialize the packet into a byte array
      uint8_t toSend[sizeof(Packet)];
      memcpy(toSend, &packet, sizeof(Packet));

      // Print and send the message
      SerialUSB.print("SENDING TEMPERATURE OF MASTER TO ALL NODES : PACKET NUMBER :");
      SerialUSB.println(packet.packetID);
      SerialUSB.println();
      SerialUSB.println();
      rf95.send(toSend, sizeof(Packet));

      SerialUSB.println();
      SerialUSB.println();

      SerialUSB.println("PRINTING EVERYONE'S TEMPERATURE");
      SerialUSB.print("(Avhi, "); SerialUSB.print(node1_temperature, 2);
      SerialUSB.print("), (Amlan, "); SerialUSB.print(node2_temperature, 2);
      SerialUSB.print("), (Shaswati, "); SerialUSB.print(node3_temperature, 2);
      SerialUSB.print("), (Anu, "); SerialUSB.print(node4_temperature, 2);
      SerialUSB.println(")");

      SerialUSB.println();
      SerialUSB.println();


  }
}


void startTimer(int frequencyHz) {
  configureClockTC4();
  configureTimerTC4(frequencyHz);
}
void configureClockTC4() {
  REG_GCLK_CLKCTRL = (uint16_t)(GCLK_CLKCTRL_CLKEN | GCLK_CLKCTRL_GEN_GCLK0 | GCLK_CLKCTRL_ID_TC4_TC5);
  while (GCLK->STATUS.bit.SYNCBUSY == 1);
}

void configureTimerTC4(int frequencyHz) {
  TcCount16* TC = (TcCount16*)TC4;
  TC->CTRLA.reg &= ~TC_CTRLA_ENABLE;
  while (TC->STATUS.bit.SYNCBUSY == 1);

  TC->CTRLA.reg |= TC_CTRLA_MODE_COUNT16 | TC_CTRLA_WAVEGEN_MFRQ | TC_CTRLA_PRESCALER_DIV1024;
  while (TC->STATUS.bit.SYNCBUSY == 1);

  setTimerFrequencyTC4(TC, frequencyHz);
  TC->INTENSET.reg = 0;
  TC->INTENSET.bit.MC0 = 1;

  NVIC_EnableIRQ(TC4_IRQn);
  TC->CTRLA.reg |= TC_CTRLA_ENABLE;
  while (TC->STATUS.bit.SYNCBUSY == 1);
}

void setTimerFrequencyTC4(TcCount16* TC, int frequencyHz) {
  int compareValue = (CPU_HZ / (TIMER_PRESCALER_DIV * frequencyHz)) - 1;
  TC->CC[0].reg = compareValue;
  while (TC->STATUS.bit.SYNCBUSY == 1);
}

void TC4_Handler() {
```

```
  static bool isLEDOn = false;
  TcCount16* TC = (TcCount16*)TC4;
  if (TC->INTFLAG.bit.MC0 == 1) {
    TC->INTFLAG.bit.MC0 = 1;
    WDT->CLEAR.reg = WDT_CLEAR_CLEAR_KEY;
//    digitalWrite(PIN_LED_13, isLEDOn);
//    SerialUSB.println(isLEDOn ? "Blue LED is on" : "Blue LED is off");
//    isLEDOn = !isLEDOn;
    canReadTemp = true;
  }
}

float Readtemp() {
  if (canReadTemp) {
    float temperature = TempZero.readInternalTemperature();

    // Remove the oldest temperature from the sum
    tempSum -= tempBuffer[tempIndex];

    // Add the new temperature to the buffer and sum
    tempBuffer[tempIndex] = temperature;
    tempSum += temperature;

    // Update the index for the oldest temperature
    tempIndex = (tempIndex + 1) % WINDOW_SIZE;

    // Update the count of temperatures added to the buffer
    if (tempCount < WINDOW_SIZE) {
      tempCount++;
    }

    // Increment the report counter
    reportCount++;

    // If 5 readings have been taken, calculate the average temperature
    if (reportCount >= WINDOW_SIZE) {
      avgTemperature = tempSum / tempCount;
      newAvgAvailable = true;  // Set the flag
      reportCount = 0;  // Reset the report counter
    }

    canReadTemp = false;

  }
  return avgTemperature;
}

void write_error(uint8_t Node_name, char* ecodeX) //---------changed
{
  int ecode=atoi(ecodeX);
  if(Node_name==3)
  {error_Amlan.write(ecode);}

   if(Node_name==2)
  {error_Shaswati.write(ecode);}

    if(Node_name==4)
  {error_Anuruddha.write(ecode);}

    if(Node_name==1)
  {error_Avhishek.write(ecode);}

    if(Node_name==5)
  {error_receive.write(ecode);}

}

void print_errors(char* err){
  SerialUSB.print("Timestamp :")
  SerialUSB.print(millis())
  SerialUSB.println("Error =  ");
  // SerialUSB.println(err);
  // SerialUSB.println(err.charAt(1));

 if (err[0] == '1') {
 SerialUSB.println("System Reset");
```

```
 }

 if (err[1] == '1') {
 SerialUSB.println("Watchdog Reset");
 }
 if (err[2] == '1') {
 SerialUSB.println("External Reset");
 }
 if (err[3] == '1') {
 SerialUSB.println("Command not allowed error: PROTECTED STATE");
 }
 if (err[4] == '1') {
 SerialUSB.println("DSU Operation Failure Error");
 }
 if (err[5] == '1') {
 SerialUSB.println("BUS ERROR Detected!");
 }
 if (err[6] == '1') {
 SerialUSB.println("COLD PLUGGING ERROR!!");
 }
 if (err[7] == '1') {
 SerialUSB.println("HOT PLUGGING ERROR!");
 }
 if (err[8] == '1') {
 SerialUSB.println("DCC1 Written Error");
 }
 if (err[9] == '1') {
 SerialUSB.println("DCC0 Written Error");
 }
 if (err[10] == '1') {
 SerialUSB.println("Debugger Probe Error");
 }
 if (err[11] == '1') {
 SerialUSB.println("Missing Packet Error!");
 }
 if (err[12] == '1') {
 SerialUSB.println("Packet Receival Error");
 }
}
```

Child Node 2

```
//SLAVE NODE = 2

# define CPU_HZ 48000000
<p class="crossnote-header " id="define-cpu_hz-48000000-2"></p>

# define TIMER_PRESCALER_DIV 1024
<p class="crossnote-header " id="define-timer_prescaler_div-1024-2"></p>

# define WINDOW_SIZE 5  // Size of the sliding window
<p class="crossnote-header " id="define-window_size-5---size-of-the-sliding-window-2"></p>

# include <TemperatureZero.h>                       //Arduino library for internal temperature measurment of the family SAMD21 and SAMD51
<p class="crossnote-header " id="include-temperaturezeroh----------arduino-library-for-internal-temperature-measurment-of-the-famil

TemperatureZero TempZero = TemperatureZero();

# include <SPI.h>                                   //This library allows you to communicate with SPI devices
<p class="crossnote-header " id="include-spih----------this-library-allows-you-to-communicate-with-spi-devices-1"></p>

# include <RH_RF95.h>
<p class="crossnote-header " id="include-rh_rf95h-1"></p>

# include <FlashStorage.h>
<p class="crossnote-header " id="include-flashstorageh-1"></p>

# include <string.h>
<p class="crossnote-header " id="include-stringh"></p>


FlashStorage(error_storage, uint16_t);
char error_bits[16];

struct Packet {
  uint8_t nodeID;
  uint16_t packetID;
  uint32_t timestamp;
  float payload;
  char error[16];
  uint8_t authID;

  // Constructor to initialize the values
  Packet() : nodeID(0), packetID(0), timestamp(0), payload(0.0), authID(0) {
    // Initialize the error array to zeros
    memset(error, '0', sizeof(error));
  }
};

volatile bool canReadTemp = false;
volatile bool Node1asked = false;


float tempBuffer[WINDOW_SIZE];  // Circular buffer to hold temperature values
int tempIndex = 0;  // Index to keep track of the oldest temperature value
float tempSum = 0;  // Sum of temperatures in the buffer
int tempCount = 0;  // Count of temperatures added to the buffer
int reportCount = 0;  // Count of readings before reporting
float avgTemperature = 0;  // To hold the average temperature
bool newAvgAvailable = false;  // Flag to indicate a new average is available

float node1_temperature = 0;
float node2_temperature = 0;
float node3_temperature = 0;
float node4_temperature = 0;


//Define functions used in the program
void startTimer(int frequencyHz);
void configureClockTC4();
void configureTimerTC4(int frequencyHz);                //
void setTimerFrequencyTC4(TcCount16* TC, int frequencyHz);
void TC4_Handler();

float Readtemp();
```

```
void write_error(String Node_name, int ecode);


int previouse_packet_id = 0;
int current_packet_id = 0;

// We need to provide the RFM95 module's chip select and interrupt pins to the
// rf95 instance below.On the SparkFun ProRF those pins are 12 and 6 respectively.
RH_RF95 rf95(12, 6);
int LED = 13; //Status LED is on pin 13
long timeSinceLastPacket = 0; //Tracks the time stamp of last packet received

float frequency = 910; //Broadcast frequency
uint16_t packetCounter = 0;  // Initialize packet counter

// long timeSinceLastPacket = 0;

void setup()
{
     GCLK->GENDIV.reg = GCLK_GENDIV_ID(2) | GCLK_GENDIV_DIV(4);
     GCLK->GENCTRL.reg = GCLK_GENCTRL_ID(2) |
     GCLK_GENCTRL_GENEN |
     GCLK_GENCTRL_SRC_OSCULP32K |
     GCLK_GENCTRL_DIVSEL;
     while(GCLK->STATUS.bit.SYNCBUSY);                              // wait while synchronizing the GCLK->STATUS.bit
                                                                    // WDT clock = clock gen 2
     GCLK->CLKCTRL.reg = GCLK_CLKCTRL_ID_WDT |
     GCLK_CLKCTRL_CLKEN |
     GCLK_CLKCTRL_GEN_GCLK2;
     WDT->CTRL.reg = 0;
     WDT->CONFIG.bit.PER = 0x8; // Set period for chip reset from the datasheet
     WDT->INTENSET.bit.EW = 1; // Enable early warning interrupt
     WDT->CTRL.bit.WEN = 0;
     SerialUSB.begin(9600);
     TempZero.init();
      // It may be difficult to read serial messages on startup. The following line
    // will wait for serial to be ready before continuing. Comment out if not needed.
    // while (!SerialUSB);
     SerialUSB.println("RFM Client!");
    //Initialize the Radio.
     if (rf95.init() == false) {
        SerialUSB.println("Radio Init Failed - Freezing");
        while (1);
        }
     else {
        //An LED inidicator to let us know radio initialization has completed.
        // rf95.setModemConfig(Bw125Cr48Sf4096); // slow and reliable?
        SerialUSB.println("Transmitter up!");
        digitalWrite(LED, HIGH);
        delay(500);
        digitalWrite(LED, LOW);
        delay(500);
        // Set frequency
     rf95.setFrequency(frequency);
        // Transmitter power can range from 14-20dbm.
     rf95.setTxPower(20, false);
     }
     pinMode(PIN_LED_13, OUTPUT);
     WDT->CTRL.bit.ENABLE = 1;
     reset_reason();
     startTimer(1);
}


void loop()
{
    avgTemperature = Readtemp();

     // if (newAvgAvailable) {
     //   // Reset the flag
     //   newAvgAvailable = false;

    if (rf95.available())
    {
        // Should be a message for us now
        uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
```

```
uint8_t len = sizeof(buf);
if (rf95.recv(buf, &len))
{
    // Cast the received buffer to the Packet struct type
    Packet *receivedPacket = (Packet *)buf;
    buf[len] = '\0';  // Null-terminate the received string
    digitalWrite(LED, HIGH); //Turn on status LED
    timeSinceLastPacket = millis(); //Timestamp this packet

    // Print the received packet details
    SerialUSB.print("Got message from Node ID: ");
    SerialUSB.println(receivedPacket->nodeID);

    // Check if this is the Master Node if so then send the Packet
    if(receivedPacket->nodeID == 1 && receivedPacket->authID == 2)
    {
      if (Node1asked == false)
      {
        Node1asked = true;
        //Print and send the average temperature
        SerialUSB.print("Average Temperature over last 5 seconds is: ");
        SerialUSB.println(avgTemperature);

        // Create the packet
        Packet packet;
        packet.nodeID = 2;  // Node 2
        packet.packetID = packetCounter++;
        packet.timestamp = millis();  // Current time in milliseconds
        packet.payload = avgTemperature;
        packet.authID  = 0;
        strcpy(packet.error,error_bits);//------------------------------------------------------------changed

        // Serialize the packet into a byte array
        uint8_t toSend[sizeof(Packet)];
        memcpy(toSend, &packet, sizeof(Packet));

        // Print and send the message
        SerialUSB.print("Node 2 Sending packet with ID: ");
        SerialUSB.println(packet.packetID);

        // SAVE NODE 1 'S TEMPERATURE
        node1_temperature = receivedPacket->payload;
        rf95.send(toSend, sizeof(Packet));


      }
    }

    else
    {  node2_temperature = avgTemperature;
      // SAVING NODE 3 AND 4 TEMPERATURES
      if(receivedPacket->nodeID == 3){
        // SAVE NODE 3 'S TEMPERATURE
        node3_temperature = receivedPacket->payload;
      }
      else if(receivedPacket->nodeID == 4){
        // SAVE NODE 4 'S TEMPERATURE
        node4_temperature = receivedPacket->payload;
      }

      Node1asked = false;
      SerialUSB.print(", Packet ID: ");
      SerialUSB.print(receivedPacket->packetID);
      SerialUSB.print(", Timestamp: ");
      SerialUSB.print(receivedPacket->timestamp);
      SerialUSB.print(", Payload (Temperature): ");
      SerialUSB.print(receivedPacket->payload);
      SerialUSB.print(" RSSI: ");
      SerialUSB.print(rf95.lastRssi(), DEC);

      SerialUSB.println();
      SerialUSB.println();

      SerialUSB.println("PRINTING EVERYONE'S TEMPERATURE");
      SerialUSB.print("(Avhi, "); SerialUSB.print(node1_temperature, 2);
      SerialUSB.print("), (Amlan, "); SerialUSB.print(node2_temperature, 2);
```

```
              SerialUSB.print("), (Shaswati, "); SerialUSB.print(node3_temperature, 2);
              SerialUSB.print("), (Anu, "); SerialUSB.print(node4_temperature, 2);
              SerialUSB.println(")");

              SerialUSB.println();
              SerialUSB.println();

              current_packet_id = receivedPacket->packetID;    //Assign the current packet ID to the current_packet_id variable

              if(current_packet_id-previouse_packet_id > 1)           //check the missing packet
              //write_error(receivedPacket->nodeID, 19) ;
              previouse_packet_id = current_packet_id;                   //save the error in the flash memory
            }
          }
         else
         {
         SerialUSB.println("Recieve failed");
        //  write_error("PKTError", 18);                              // Save the error code as message receive failed
         }
     }
     timeSinceLastPacket = millis();
 }



void startTimer(int frequencyHz) {
  configureClockTC4();
  configureTimerTC4(frequencyHz);
}
void configureClockTC4() {
  REG_GCLK_CLKCTRL = (uint16_t)(GCLK_CLKCTRL_CLKEN | GCLK_CLKCTRL_GEN_GCLK0 | GCLK_CLKCTRL_ID_TC4_TC5);
  while (GCLK->STATUS.bit.SYNCBUSY == 1);
}

void configureTimerTC4(int frequencyHz) {
  TcCount16* TC = (TcCount16*)TC4;
  TC->CTRLA.reg &= ~TC_CTRLA_ENABLE;
  while (TC->STATUS.bit.SYNCBUSY == 1);

  TC->CTRLA.reg |= TC_CTRLA_MODE_COUNT16 | TC_CTRLA_WAVEGEN_MFRQ | TC_CTRLA_PRESCALER_DIV1024;
  while (TC->STATUS.bit.SYNCBUSY == 1);

  setTimerFrequencyTC4(TC, frequencyHz);
  TC->INTENSET.reg = 0;
  TC->INTENSET.bit.MC0 = 1;

  NVIC_EnableIRQ(TC4_IRQn);
  TC->CTRLA.reg |= TC_CTRLA_ENABLE;
  while (TC->STATUS.bit.SYNCBUSY == 1);
}

void setTimerFrequencyTC4(TcCount16* TC, int frequencyHz) {
  int compareValue = (CPU_HZ / (TIMER_PRESCALER_DIV * frequencyHz)) - 1;
  TC->CC[0].reg = compareValue;
  while (TC->STATUS.bit.SYNCBUSY == 1);
}

void TC4_Handler()
{
    static bool isLEDOn = false;
    TcCount16* TC = (TcCount16*)TC4;
    if (TC->INTFLAG.bit.MC0 == 1)
    {
        TC->INTFLAG.bit.MC0 = 1;
        WDT->CLEAR.reg = WDT_CLEAR_CLEAR_KEY;
    //    digitalWrite(PIN_LED_13, isLEDOn);
    //    SerialUSB.println(isLEDOn ? "Blue LED is on" : "Blue LED is off");
    //    isLEDOn = !isLEDOn;
        canReadTemp = true;
    }
}

float Readtemp() {
  if (canReadTemp) {
    float temperature = TempZero.readInternalTemperature();
```

```
    // Remove the oldest temperature from the sum
    tempSum -= tempBuffer[tempIndex];

    // Add the new temperature to the buffer and sum
    tempBuffer[tempIndex] = temperature;
    tempSum += temperature;

    // Update the index for the oldest temperature
    tempIndex = (tempIndex + 1) % WINDOW_SIZE;

    // Update the count of temperatures added to the buffer
    if (tempCount < WINDOW_SIZE) {
      tempCount++;
    }

    // Increment the report counter
    reportCount++;

    // If 5 readings have been taken, calculate the average temperature
    if (reportCount >= WINDOW_SIZE) {
      avgTemperature = tempSum / tempCount;
      newAvgAvailable = true;  // Set the flag
      reportCount = 0;  // Reset the report counter
    }

    canReadTemp = false;
  }
  return avgTemperature;
}


void reset_reason(){
 //determine last reset type
 uint8_t reset_reg = PM->RCAUSE.reg;
 SerialUSB.println(bitRead(reset_reg, 6));
 if(bitRead(reset_reg, 6)) {error_bits[0]=1;}//-------------------------------------------changed
 if(bitRead(reset_reg, 5)) {error_bits[1]=1;}
 if(bitRead(reset_reg, 4)) {error_bits[2]=1;}
//  bitWrite(error_bits, 0, bitRead(reset_reg, 6));
//  bitWrite(error_bits, 1, bitRead(reset_reg, 5));
//  bitWrite(error_bits, 2, bitRead(reset_reg, 4));
//   bitWrite(error_bits, 13, 1);
}

void WDT_Handler() {
    //SerialUSB.println("WDT Interrupt");
    uint8_t reg_statusA = REG_DSU_STATUSA;
    uint8_t reg_statusB = REG_DSU_STATUSB;
    uint16_t status_errors = 0x0000;
    status_errors = reg_statusA;
    status_errors = status_errors << 8;
    status_errors |= reg_statusB;

    if(bitRead(status_errors, 12)) {error_bits[3]=1;}
    if(bitRead(status_errors, 11)) {error_bits[4]=1;}
    if(bitRead(status_errors, 10)) {error_bits[5]=1;}
    if(bitRead(status_errors, 9)) {error_bits[6]=1;}
    if(bitRead(status_errors, 4)) {error_bits[7]=1;}
    if(bitRead(status_errors, 3)) {error_bits[8]=1;}
    if(bitRead(status_errors, 2)) {error_bits[9]=1;}
    if(bitRead(status_errors, 1)) {error_bits[10]=1;}
}
```

Child Node 3

```
//SLAVE NODE = 3

# define CPU_HZ 48000000
<p class="crossnote-header " id="define-cpu_hz-48000000-3"></p>

# define TIMER_PRESCALER_DIV 1024
<p class="crossnote-header " id="define-timer_prescaler_div-1024-3"></p>

# define WINDOW_SIZE 5  // Size of the sliding window
<p class="crossnote-header " id="define-window_size-5---size-of-the-sliding-window-3"></p>

# include <TemperatureZero.h>                       //Arduino library for internal temperature measurment of the family SAMD21 and SAMD51
<p class="crossnote-header " id="include-temperaturezeroh----------arduino-library-for-internal-temperature-measurment-of-the-famil

TemperatureZero TempZero = TemperatureZero();

# include <SPI.h>                                   //This library allows you to communicate with SPI devices
<p class="crossnote-header " id="include-spih----------this-library-allows-you-to-communicate-with-spi-devices-2"></p>

# include <RH_RF95.h>
<p class="crossnote-header " id="include-rh_rf95h-2"></p>

# include <FlashStorage.h>
<p class="crossnote-header " id="include-flashstorageh-2"></p>

# include <string.h>
<p class="crossnote-header " id="include-stringh-1"></p>


FlashStorage(error_storage, uint16_t);
char error_bits[16];

struct Packet {
  uint8_t nodeID;
  uint16_t packetID;
  uint32_t timestamp;
  float payload;
  char error[16];
  uint8_t authID;

  // Constructor to initialize the values
  Packet() : nodeID(0), packetID(0), timestamp(0), payload(0.0), authID(0) {
    // Initialize the error array to zeros
    memset(error, '0', sizeof(error));
  }
};

volatile bool canReadTemp = false;
volatile bool Node1asked = false;


float tempBuffer[WINDOW_SIZE];  // Circular buffer to hold temperature values
int tempIndex = 0;  // Index to keep track of the oldest temperature value
float tempSum = 0;  // Sum of temperatures in the buffer
int tempCount = 0;  // Count of temperatures added to the buffer
int reportCount = 0;  // Count of readings before reporting
float avgTemperature = 0;  // To hold the average temperature
bool newAvgAvailable = false;  // Flag to indicate a new average is available

float node1_temperature = 0;
float node2_temperature = 0;
float node3_temperature = 0;
float node4_temperature = 0;

//Define functions used in the program
void startTimer(int frequencyHz);
void configureClockTC4();
void configureTimerTC4(int frequencyHz);                //
void setTimerFrequencyTC4(TcCount16* TC, int frequencyHz);
void TC4_Handler();

float Readtemp();
void write_error(String Node_name, int ecode);
```

```
int previouse_packet_id = 0;
int current_packet_id = 0;

// We need to provide the RFM95 module's chip select and interrupt pins to the
// rf95 instance below.On the SparkFun ProRF those pins are 12 and 6 respectively.
RH_RF95 rf95(12, 6);
int LED = 13; //Status LED is on pin 13
long timeSinceLastPacket = 0; //Tracks the time stamp of last packet received

float frequency = 910; //Broadcast frequency
uint16_t packetCounter = 0;  // Initialize packet counter

// long timeSinceLastPacket = 0;

void setup()
{
     GCLK->GENDIV.reg = GCLK_GENDIV_ID(2) | GCLK_GENDIV_DIV(4);
     GCLK->GENCTRL.reg = GCLK_GENCTRL_ID(2) |
     GCLK_GENCTRL_GENEN |
     GCLK_GENCTRL_SRC_OSCULP32K |
     GCLK_GENCTRL_DIVSEL;
     while(GCLK->STATUS.bit.SYNCBUSY);                              // wait while synchronizing the GCLK->STATUS.bit
                                                                    // WDT clock = clock gen 2
     GCLK->CLKCTRL.reg = GCLK_CLKCTRL_ID_WDT |
     GCLK_CLKCTRL_CLKEN |
     GCLK_CLKCTRL_GEN_GCLK2;
     WDT->CTRL.reg = 0;
     WDT->CONFIG.bit.PER = 0x8; // Set period for chip reset from the datasheet
     WDT->INTENSET.bit.EW = 1; // Enable early warning interrupt
     WDT->CTRL.bit.WEN = 0;
     SerialUSB.begin(9600);
     TempZero.init();
      // It may be difficult to read serial messages on startup. The following line
    // will wait for serial to be ready before continuing. Comment out if not needed.
    // while (!SerialUSB);
     SerialUSB.println("RFM Client!");
    //Initialize the Radio.
     if (rf95.init() == false) {
        SerialUSB.println("Radio Init Failed - Freezing");
        while (1);
        }
     else {
        //An LED inidicator to let us know radio initialization has completed.
        // rf95.setModemConfig(Bw125Cr48Sf4096); // slow and reliable?
        SerialUSB.println("Transmitter up!");
        digitalWrite(LED, HIGH);
        delay(500);
        digitalWrite(LED, LOW);
        delay(500);
        // Set frequency
     rf95.setFrequency(frequency);
        // Transmitter power can range from 14-20dbm.
     rf95.setTxPower(20, false);
     }
     pinMode(PIN_LED_13, OUTPUT);
     WDT->CTRL.bit.ENABLE = 1;
     reset_reason();
     startTimer(1);
}


void loop()
{
    avgTemperature = Readtemp();

     // if (newAvgAvailable) {
     //   // Reset the flag
     //   newAvgAvailable = false;

    if (rf95.available())
    {
        // Should be a message for us now
        uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
        uint8_t len = sizeof(buf);
```

```
if (rf95.recv(buf, &len))
{
    // Cast the received buffer to the Packet struct type
    Packet *receivedPacket = (Packet *)buf;
    buf[len] = '\0';  // Null-terminate the received string
    digitalWrite(LED, HIGH); //Turn on status LED
    timeSinceLastPacket = millis(); //Timestamp this packet

    // Print the received packet details
    SerialUSB.print("Got message from Node ID: ");
    SerialUSB.println(receivedPacket->nodeID);

    // Check if this is the Master Node if so then send the Packet
    if(receivedPacket->nodeID == 1 && receivedPacket->authID == 3)
    {
      if (Node1asked == false)
      {
        Node1asked = true;
        //Print and send the average temperature
        SerialUSB.print("Average Temperature over last 5 seconds is: ");
        SerialUSB.println(avgTemperature);

        // Create the packet
        Packet packet;
        packet.nodeID = 3;  // Node 3
        packet.packetID = packetCounter++;
        packet.timestamp = millis();  // Current time in milliseconds
        packet.payload = avgTemperature;
        packet.authID  = 0;
        strcpy(packet.error,error_bits);//------------------------------------------------------------changed

        // Serialize the packet into a byte array
        uint8_t toSend[sizeof(Packet)];
        memcpy(toSend, &packet, sizeof(Packet));

        // Print and send the message
        SerialUSB.print("Node 3 Sending packet with ID: ");
        SerialUSB.println(packet.packetID);

        rf95.send(toSend, sizeof(Packet));

        // SAVE NODE 1 'S TEMPERATURE
        node1_temperature = receivedPacket->payload;
      }
    }

    else
    { node3_temperature = avgTemperature;
      // SAVING NODE 2 AND 4 TEMPERATURES
      if(receivedPacket->nodeID == 2){
        // SAVE NODE 2 'S TEMPERATURE
        node2_temperature = receivedPacket->payload;
      }
      else if(receivedPacket->nodeID == 4){
        // SAVE NODE 4 'S TEMPERATURE
        node4_temperature = receivedPacket->payload;
      }
      Node1asked = false;
      // SerialUSB.print(", Packet ID: ");
      // SerialUSB.print(receivedPacket->packetID);
      // SerialUSB.print(", Timestamp: ");
      // SerialUSB.print(receivedPacket->timestamp);
      // SerialUSB.print(", Payload (Temperature): ");
      // SerialUSB.print(receivedPacket->payload);
      // SerialUSB.print(" RSSI: ");
      // SerialUSB.print(rf95.lastRssi(), DEC);
      // SerialUSB.println();

      SerialUSB.println();
      SerialUSB.println();

      SerialUSB.println("PRINTING EVERYONE'S TEMPERATURE");
      SerialUSB.print("(Avhi, "); SerialUSB.print(node1_temperature, 2);
      SerialUSB.print("), (Amlan, "); SerialUSB.print(node2_temperature, 2);
      SerialUSB.print("), (Shaswati, "); SerialUSB.print(node3_temperature, 2);
      SerialUSB.print("), (Anu, "); SerialUSB.print(node4_temperature, 2);
```

```
              SerialUSB.println(")");

              SerialUSB.println();
              SerialUSB.println();


              current_packet_id = receivedPacket->packetID;    //Assign the current packet ID to the current_packet_id variable

              if(current_packet_id-previouse_packet_id > 1)            //check the missing packet
              //write_error(receivedPacket->nodeID, 19) ;
              previouse_packet_id = current_packet_id;                   //save the error in the flash memory
             }
           }
          else
          {
          SerialUSB.println("Recieve failed");
         //  write_error("PKTError", 18);                              // Save the error code as message receive failed
          }
     }
     timeSinceLastPacket = millis();
 }



void startTimer(int frequencyHz) {
  configureClockTC4();
  configureTimerTC4(frequencyHz);
}
void configureClockTC4() {
  REG_GCLK_CLKCTRL = (uint16_t)(GCLK_CLKCTRL_CLKEN | GCLK_CLKCTRL_GEN_GCLK0 | GCLK_CLKCTRL_ID_TC4_TC5);
  while (GCLK->STATUS.bit.SYNCBUSY == 1);
}

void configureTimerTC4(int frequencyHz) {
  TcCount16* TC = (TcCount16*)TC4;
  TC->CTRLA.reg &= ~TC_CTRLA_ENABLE;
  while (TC->STATUS.bit.SYNCBUSY == 1);

  TC->CTRLA.reg |= TC_CTRLA_MODE_COUNT16 | TC_CTRLA_WAVEGEN_MFRQ | TC_CTRLA_PRESCALER_DIV1024;
  while (TC->STATUS.bit.SYNCBUSY == 1);

  setTimerFrequencyTC4(TC, frequencyHz);
  TC->INTENSET.reg = 0;
  TC->INTENSET.bit.MC0 = 1;

  NVIC_EnableIRQ(TC4_IRQn);
  TC->CTRLA.reg |= TC_CTRLA_ENABLE;
  while (TC->STATUS.bit.SYNCBUSY == 1);
}

void setTimerFrequencyTC4(TcCount16* TC, int frequencyHz) {
  int compareValue = (CPU_HZ / (TIMER_PRESCALER_DIV * frequencyHz)) - 1;
  TC->CC[0].reg = compareValue;
  while (TC->STATUS.bit.SYNCBUSY == 1);
}

void TC4_Handler()
{
    static bool isLEDOn = false;
    TcCount16* TC = (TcCount16*)TC4;
    if (TC->INTFLAG.bit.MC0 == 1)
    {
        TC->INTFLAG.bit.MC0 = 1;
        WDT->CLEAR.reg = WDT_CLEAR_CLEAR_KEY;
    //    digitalWrite(PIN_LED_13, isLEDOn);
    //    SerialUSB.println(isLEDOn ? "Blue LED is on" : "Blue LED is off");
    //    isLEDOn = !isLEDOn;
        canReadTemp = true;
    }
}

float Readtemp() {
  if (canReadTemp) {
    float temperature = TempZero.readInternalTemperature();

    // Remove the oldest temperature from the sum
```

```
    tempSum -= tempBuffer[tempIndex];

    // Add the new temperature to the buffer and sum
    tempBuffer[tempIndex] = temperature;
    tempSum += temperature;

    // Update the index for the oldest temperature
    tempIndex = (tempIndex + 1) % WINDOW_SIZE;

    // Update the count of temperatures added to the buffer
    if (tempCount < WINDOW_SIZE) {
      tempCount++;
    }

    // Increment the report counter
    reportCount++;

    // If 5 readings have been taken, calculate the average temperature
    if (reportCount >= WINDOW_SIZE) {
      avgTemperature = tempSum / tempCount;
      newAvgAvailable = true;  // Set the flag
      reportCount = 0;  // Reset the report counter
    }

    canReadTemp = false;
  }
  return avgTemperature;
}


void reset_reason(){
 //determine last reset type
 uint8_t reset_reg = PM->RCAUSE.reg;
 SerialUSB.println(bitRead(reset_reg, 6));
 if(bitRead(reset_reg, 6)) {error_bits[0]=1;}//-------------------------------------------changed
 if(bitRead(reset_reg, 5)) {error_bits[1]=1;}
 if(bitRead(reset_reg, 4)) {error_bits[2]=1;}
//  bitWrite(error_bits, 0, bitRead(reset_reg, 6));
//  bitWrite(error_bits, 1, bitRead(reset_reg, 5));
//  bitWrite(error_bits, 2, bitRead(reset_reg, 4));
//   bitWrite(error_bits, 13, 1);
}

void WDT_Handler() {
    //SerialUSB.println("WDT Interrupt");
    uint8_t reg_statusA = REG_DSU_STATUSA;
    uint8_t reg_statusB = REG_DSU_STATUSB;
    uint16_t status_errors = 0x0000;
    status_errors = reg_statusA;
    status_errors = status_errors << 8;
    status_errors |= reg_statusB;

    if(bitRead(status_errors, 12)) {error_bits[3]=1;}
    if(bitRead(status_errors, 11)) {error_bits[4]=1;}
    if(bitRead(status_errors, 10)) {error_bits[5]=1;}
    if(bitRead(status_errors, 9)) {error_bits[6]=1;}
    if(bitRead(status_errors, 4)) {error_bits[7]=1;}
    if(bitRead(status_errors, 3)) {error_bits[8]=1;}
    if(bitRead(status_errors, 2)) {error_bits[9]=1;}
    if(bitRead(status_errors, 1)) {error_bits[10]=1;}
}
```

Child Node 4

```
//SLAVE NODE = 4

# define CPU_HZ 48000000
<p class="crossnote-header " id="define-cpu_hz-48000000-4"></p>

# define TIMER_PRESCALER_DIV 1024
<p class="crossnote-header " id="define-timer_prescaler_div-1024-4"></p>

# define WINDOW_SIZE 5  // Size of the sliding window
<p class="crossnote-header " id="define-window_size-5---size-of-the-sliding-window-4"></p>

# include <TemperatureZero.h>                       //Arduino library for internal temperature measurment of the family SAMD21 and SAMD51
<p class="crossnote-header " id="include-temperaturezeroh----------arduino-library-for-internal-temperature-measurment-of-the-famil

TemperatureZero TempZero = TemperatureZero();

# include <SPI.h>                                   //This library allows you to communicate with SPI devices
<p class="crossnote-header " id="include-spih----------this-library-allows-you-to-communicate-with-spi-devices-3"></p>

# include <RH_RF95.h>
<p class="crossnote-header " id="include-rh_rf95h-3"></p>

# include <FlashStorage.h>
<p class="crossnote-header " id="include-flashstorageh-3"></p>

# include <string.h>
<p class="crossnote-header " id="include-stringh-2"></p>


FlashStorage(error_storage, uint16_t);
char error_bits[16];

struct Packet {
  uint8_t nodeID;
  uint16_t packetID;
  uint32_t timestamp;
  float payload;
  char error[16];
  uint8_t authID;

  // Constructor to initialize the values
  Packet() : nodeID(0), packetID(0), timestamp(0), payload(0.0), authID(0) {
    // Initialize the error array to zeros
    memset(error, '0', sizeof(error));
  }
};

volatile bool canReadTemp = false;
volatile bool Node1asked = false;


float tempBuffer[WINDOW_SIZE];  // Circular buffer to hold temperature values
int tempIndex = 0;  // Index to keep track of the oldest temperature value
float tempSum = 0;  // Sum of temperatures in the buffer
int tempCount = 0;  // Count of temperatures added to the buffer
int reportCount = 0;  // Count of readings before reporting
float avgTemperature = 0;  // To hold the average temperature
bool newAvgAvailable = false;  // Flag to indicate a new average is available

float node1_temperature = 0;
float node2_temperature = 0;
float node3_temperature = 0;
float node4_temperature = 0;




//Define functions used in the program
void startTimer(int frequencyHz);
void configureClockTC4();
void configureTimerTC4(int frequencyHz);                //
void setTimerFrequencyTC4(TcCount16* TC, int frequencyHz);
void TC4_Handler();
```

```
float Readtemp();
void write_error(String Node_name, int ecode);

int previouse_packet_id = 0;
int current_packet_id = 0;



// We need to provide the RFM95 module's chip select and interrupt pins to the
// rf95 instance below.On the SparkFun ProRF those pins are 12 and 6 respectively.
RH_RF95 rf95(12, 6);
int LED = 13; //Status LED is on pin 13
long timeSinceLastPacket = 0; //Tracks the time stamp of last packet received

float frequency = 910; //Broadcast frequency
uint16_t packetCounter = 0;  // Initialize packet counter

// long timeSinceLastPacket = 0;

void setup()
{
     GCLK->GENDIV.reg = GCLK_GENDIV_ID(2) | GCLK_GENDIV_DIV(4);
     GCLK->GENCTRL.reg = GCLK_GENCTRL_ID(2) |
     GCLK_GENCTRL_GENEN |
     GCLK_GENCTRL_SRC_OSCULP32K |
     GCLK_GENCTRL_DIVSEL;
     while(GCLK->STATUS.bit.SYNCBUSY);                              // wait while synchronizing the GCLK->STATUS.bit
                                                                    // WDT clock = clock gen 2
     GCLK->CLKCTRL.reg = GCLK_CLKCTRL_ID_WDT |
     GCLK_CLKCTRL_CLKEN |
     GCLK_CLKCTRL_GEN_GCLK2;
     WDT->CTRL.reg = 0;
     WDT->CONFIG.bit.PER = 0x8; // Set period for chip reset from the datasheet
     WDT->INTENSET.bit.EW = 1; // Enable early warning interrupt
     WDT->CTRL.bit.WEN = 0;
     SerialUSB.begin(9600);
     TempZero.init();
      // It may be difficult to read serial messages on startup. The following line
    // will wait for serial to be ready before continuing. Comment out if not needed.
    // while (!SerialUSB);
     SerialUSB.println("RFM Client!");
    //Initialize the Radio.
     if (rf95.init() == false) {
        SerialUSB.println("Radio Init Failed - Freezing");
        while (1);
        }
     else {
        //An LED inidicator to let us know radio initialization has completed.
        // rf95.setModemConfig(Bw125Cr48Sf4096); // slow and reliable?
        SerialUSB.println("Transmitter up!");
        digitalWrite(LED, HIGH);
        delay(500);
        digitalWrite(LED, LOW);
        delay(500);
        // Set frequency
     rf95.setFrequency(frequency);
        // Transmitter power can range from 14-20dbm.
     rf95.setTxPower(20, false);
     }
     pinMode(PIN_LED_13, OUTPUT);
     WDT->CTRL.bit.ENABLE = 1;
     reset_reason();
     startTimer(1);
}


void loop()
{
    avgTemperature = Readtemp();

     // if (newAvgAvailable) {
     //   // Reset the flag
     //   newAvgAvailable = false;

    if (rf95.available())
```

```
{
    // Should be a message for us now
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
    if (rf95.recv(buf, &len))
    {
        // Cast the received buffer to the Packet struct type
        Packet *receivedPacket = (Packet *)buf;
        buf[len] = '\0';  // Null-terminate the received string
        digitalWrite(LED, HIGH); //Turn on status LED
        timeSinceLastPacket = millis(); //Timestamp this packet

        // Print the received packet details
        SerialUSB.print("Got message from Node ID: ");
        SerialUSB.println(receivedPacket->nodeID);

        // Check if this is the Master Node if so then send the Packet
        if(receivedPacket->nodeID == 1 && receivedPacket->authID == 4)
        {
          if (Node1asked == false)
          {
            Node1asked = true;
            //Print and send the average temperature
            SerialUSB.print("Average Temperature over last 5 seconds is: ");
            SerialUSB.println(avgTemperature);

            // Create the packet
            Packet packet;
            packet.nodeID = 4;  // Node 4
            packet.packetID = packetCounter++;
            packet.timestamp = millis();  // Current time in milliseconds
            packet.payload = avgTemperature;
            packet.authID  = 0;
            strcpy(packet.error,error_bits);//------------------------------------------------------------changed

            // Serialize the packet into a byte array
            uint8_t toSend[sizeof(Packet)];
            memcpy(toSend, &packet, sizeof(Packet));

            // Print and send the message
            SerialUSB.print("Node 4 Sending packet with ID: ");
            SerialUSB.println(packet.packetID);

            rf95.send(toSend, sizeof(Packet));

            // SAVE NODE 1 'S TEMPERATURE
            node1_temperature = receivedPacket->payload;
          }
        }

        else
        {
          node4_temperature = avgTemperature;
          // SAVING NODE 2 AND 3 TEMPERATURES
          if(receivedPacket->nodeID == 2){
            // SAVE NODE 2 'S TEMPERATURE
            node2_temperature = receivedPacket->payload;
          }
          else if(receivedPacket->nodeID == 3){
            // SAVE NODE 3 'S TEMPERATURE
            node3_temperature = receivedPacket->payload;
          }
          Node1asked = false;
          // SerialUSB.print(", Packet ID: ");
          // SerialUSB.print(receivedPacket->packetID);
          // SerialUSB.print(", Timestamp: ");
          // SerialUSB.print(receivedPacket->timestamp);
          // SerialUSB.print(", Payload (Temperature): ");
          // SerialUSB.print(receivedPacket->payload);
          // SerialUSB.print(" RSSI: ");
          // SerialUSB.print(rf95.lastRssi(), DEC);
          // SerialUSB.println();

          SerialUSB.println();
          SerialUSB.println();
```

```
              SerialUSB.println("PRINTING EVERYONE'S TEMPERATURE");
              SerialUSB.print("(Avhi, "); SerialUSB.print(node1_temperature, 2);
              SerialUSB.print("), (Amlan, "); SerialUSB.print(node2_temperature, 2);
              SerialUSB.print("), (Shaswati, "); SerialUSB.print(node3_temperature, 2);
              SerialUSB.print("), (Anu, "); SerialUSB.print(node4_temperature, 2);
              SerialUSB.println(")");

              SerialUSB.println();
              SerialUSB.println();
              // delay(500);
              current_packet_id = receivedPacket->packetID;    //Assign the current packet ID to the current_packet_id variable

              if(current_packet_id-previouse_packet_id > 1)            //check the missing packet
              //write_error(receivedPacket->nodeID, 19) ;
              previouse_packet_id = current_packet_id;                   //save the error in the flash memory
            }
          }
         else
         {
         SerialUSB.println("Recieve failed");
        //  write_error("PKTError", 18);                              // Save the error code as message receive failed
         }
     }
     timeSinceLastPacket = millis();
 }



void startTimer(int frequencyHz) {
  configureClockTC4();
  configureTimerTC4(frequencyHz);
}
void configureClockTC4() {
  REG_GCLK_CLKCTRL = (uint16_t)(GCLK_CLKCTRL_CLKEN | GCLK_CLKCTRL_GEN_GCLK0 | GCLK_CLKCTRL_ID_TC4_TC5);
  while (GCLK->STATUS.bit.SYNCBUSY == 1);
}

void configureTimerTC4(int frequencyHz) {
  TcCount16* TC = (TcCount16*)TC4;
  TC->CTRLA.reg &= ~TC_CTRLA_ENABLE;
  while (TC->STATUS.bit.SYNCBUSY == 1);

  TC->CTRLA.reg |= TC_CTRLA_MODE_COUNT16 | TC_CTRLA_WAVEGEN_MFRQ | TC_CTRLA_PRESCALER_DIV1024;
  while (TC->STATUS.bit.SYNCBUSY == 1);

  setTimerFrequencyTC4(TC, frequencyHz);
  TC->INTENSET.reg = 0;
  TC->INTENSET.bit.MC0 = 1;

  NVIC_EnableIRQ(TC4_IRQn);
  TC->CTRLA.reg |= TC_CTRLA_ENABLE;
  while (TC->STATUS.bit.SYNCBUSY == 1);
}

void setTimerFrequencyTC4(TcCount16* TC, int frequencyHz) {
  int compareValue = (CPU_HZ / (TIMER_PRESCALER_DIV * frequencyHz)) - 1;
  TC->CC[0].reg = compareValue;
  while (TC->STATUS.bit.SYNCBUSY == 1);
}

void TC4_Handler()
{
    static bool isLEDOn = false;
    TcCount16* TC = (TcCount16*)TC4;
    if (TC->INTFLAG.bit.MC0 == 1)
    {
        TC->INTFLAG.bit.MC0 = 1;
        WDT->CLEAR.reg = WDT_CLEAR_CLEAR_KEY;
    //    digitalWrite(PIN_LED_13, isLEDOn);
    //    SerialUSB.println(isLEDOn ? "Blue LED is on" : "Blue LED is off");
    //    isLEDOn = !isLEDOn;
        canReadTemp = true;
    }
}

float Readtemp() {
```

```
  if (canReadTemp) {
    float temperature = TempZero.readInternalTemperature();

    // Remove the oldest temperature from the sum
    tempSum -= tempBuffer[tempIndex];

    // Add the new temperature to the buffer and sum
    tempBuffer[tempIndex] = temperature;
    tempSum += temperature;

    // Update the index for the oldest temperature
    tempIndex = (tempIndex + 1) % WINDOW_SIZE;

    // Update the count of temperatures added to the buffer
    if (tempCount < WINDOW_SIZE) {
      tempCount++;
    }

    // Increment the report counter
    reportCount++;

    // If 5 readings have been taken, calculate the average temperature
    if (reportCount >= WINDOW_SIZE) {
      avgTemperature = tempSum / tempCount;
      newAvgAvailable = true;  // Set the flag
      reportCount = 0;  // Reset the report counter
    }

    canReadTemp = false;
  }
  return avgTemperature;
}


void reset_reason(){
 //determine last reset type
 uint8_t reset_reg = PM->RCAUSE.reg;
 SerialUSB.println(bitRead(reset_reg, 6));
 if(bitRead(reset_reg, 6)) {error_bits[0]=1;}//-----------------------------------------changed
 if(bitRead(reset_reg, 5)) {error_bits[1]=1;}
 if(bitRead(reset_reg, 4)) {error_bits[2]=1;}
//  bitWrite(error_bits, 0, bitRead(reset_reg, 6));
//  bitWrite(error_bits, 1, bitRead(reset_reg, 5));
//  bitWrite(error_bits, 2, bitRead(reset_reg, 4));
//   bitWrite(error_bits, 13, 1);
}

void WDT_Handler() {
    //SerialUSB.println("WDT Interrupt");
    uint8_t reg_statusA = REG_DSU_STATUSA;
    uint8_t reg_statusB = REG_DSU_STATUSB;
    uint16_t status_errors = 0x0000;
    status_errors = reg_statusA;
    status_errors = status_errors << 8;
    status_errors |= reg_statusB;

    if(bitRead(status_errors, 12)) {error_bits[3]=1;}
    if(bitRead(status_errors, 11)) {error_bits[4]=1;}
    if(bitRead(status_errors, 10)) {error_bits[5]=1;}
    if(bitRead(status_errors, 9)) {error_bits[6]=1;}
    if(bitRead(status_errors, 4)) {error_bits[7]=1;}
    if(bitRead(status_errors, 3)) {error_bits[8]=1;}
    if(bitRead(status_errors, 2)) {error_bits[9]=1;}
    if(bitRead(status_errors, 1)) {error_bits[10]=1;}
}
```

References:

The refernces are as follows:

1. Arduino Official Documentation
2. Adafruit Learning Resource

3. Noergaard, Tammy. *Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers.* Newnes, 2012.
4. Davies, John H. *MSP430 Microcontroller Basics.* Elsevier, 2008.