Lab 6 Solution: Programming and Remote Control of a DISAL Arduino Xbee Sensor Node

Part 1: Control remote sensor

1. **(I):** Open the code in *part1/basestation* and complete the C code to modify the variable *command* based on which button is pressed. Depending on which button is pressed, the user will ask for temperature, humidity, or light data. Specifically:
   - If BTN_T is pressed, command=T
   - If BTN_H is pressed, command =H
   - If BTN_L is pressed, command =L

   Solution:

```
if (digitalRead(BTN_T) == false)
{
     command="T";
}else if (digitalRead(BTN_H) == false)
{
     command="H";
}else if (digitalRead(BTN_L) == false){
     command="L";
}
```

2. **(S):** Now look at the whole code. Can you explain exactly what is happening? When is the command for new data sent to the remote sensor? Upload the code to your board

   Solution: In the setup() function, we initialize the various components of the system. In the loop() function, the code checks if the user is pressing one of the three buttons and records the user preference on the variable "command". If the command changes from the previous iteration (because the user has pressed a button) the new command is sent to the remote sensor, and it is then saved in a new variable to keep track of the history of the commands. The last part of the code checks if there is any data coming from the remote node. If there is, the data is sent to the computer on the Serial line.

3. **(I):** Open the code in *part1/remotesensor* and complete the C code to send back the remaining two types of data when requested by the receiver. Be careful about maintaining the format "LETTER DATA" (where letter is L, H, T and data is the respective data) for all the data you send back. *Hint: Use the functions getHumidity() and getTemperature.*

   Solution:

```
if (data=="T"){
     float t = getTemperature();
     Serial3.println("T "+String(t));
```

```
        oled.print(String(t));
    }else if (data=="H"){
        float h = getHumidity();
        Serial3.println("H "+String(h));
        oled.print(String(h));
    }else if (data=="L"){
        Tsl.on();
        Tsl.setSensitivity(true, Tsl2561::EXP_14);
        delay(16);
        uint8_t id;
        uint16_t full;
        Tsl.id(id);
        Tsl.fullLuminosity(full);
        Tsl.off();
        Serial3.println("L "+String(full));
        oled.print(String(full));
    }
```

4. **(S):** With the boards turned on, run the scripts *saveSerialData.py* and *cleanData.py* located in the *python* folder, similarly to what you did in the last lab. Press the buttons while you save the data to collect different types of data.

   Solution: Check your implementation

5. **(I):** Open the *plotter.m* script contained in the *matlab* folder. Write MATLAB code to compute the average, min and max of the three quantities (light, humidity and temperature) using the data that you saved.

   Solution: Sample code to compute the average of light values. Change the value of the variable type from L to H or T to compute temperature or humidity

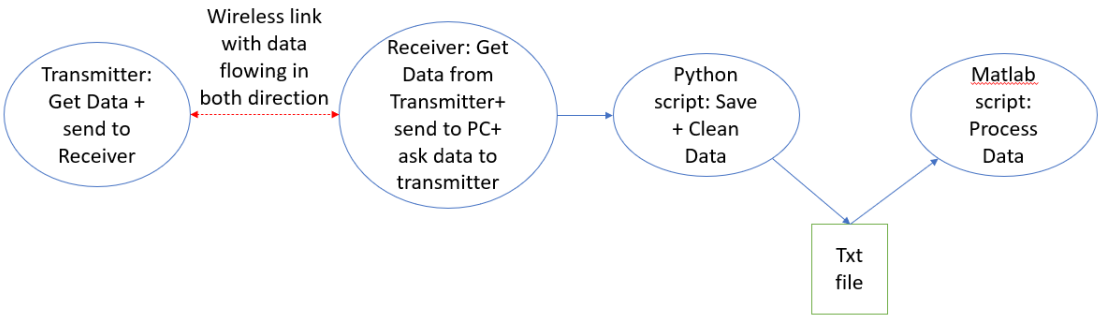
```
type='L';
avg=0;
counter=0;
max=0; %we know that no value is less than 0 in this
scenario
min= intmax;
for i=1:height(T)
    if string(T.Var1(i))==type
        avg=avg+T.Var2(i);
        counter=counter+1;
        %check for max
        if T.Var2(i)>max
            max=T.Var2(i);
        end
        %check for min
        if T.Var2(i)<min
            min=T.Var2(i);
        end
    end
```

```
end

if counter>0
    avg =avg/counter;
else
    disp("No value of type "+type);
end
```

6. **(Q):** You just worked on a more complex data sharing network. Can you draw a schematic of the data sharing and the data processing that you just implemented?

   Solution:



Part 2: Interrupts

7. **(I):** Go to the folder *part2/environmentalmonitoring* and look at the code. Complete the C code to keep track of the number of times the button is pressed.

   Solution:

```
if (digitalRead(BTN_UP) == false)
   {
    button_press++;
   }
```

8. **(S):** Test your code by pressing the button closer to the screen. What is happening? Why?

   Solution: The value of the variable button_press is increased only if the button is pressed when the line of code written for question I7 is executed. This happens once every 10 seconds. If the button is pressed while other lines of code are executed, the button press will not be recorded.

9. **(I)**: Write C code to increase the value of the variable "value" in the allocated space and test your code by opening the serial monitor and pressing the button closer to the shield's screen to increase the value of the printed variable. How does the value increase? Why?

Solution:

```
ISR(PCINT0_vect)
{
// YOUR CODE HERE ////////////
value++;
//////////////////////////////
}
```

The value increases by 2 for each button press because it increases for every change in the state of the button (once when the button is pressed, once when it is released). In fact, the interrupt is "event triggered" and it is called for both falling and rising edges of the button signal. If you keep the button pressed, you will see that the value changes by 1 number.

10. **(S):** Go back to the code. Do you notice something unusual about the usage of the function ISR(PCINT0_vect)? Can you guess what is happening?

    Solution: This function is not called anywhere in the code. This is because this function is a special function that is already "known" by the microcontroller. Because of this, both the name and the arguments of the function cannot be changed.

11. **(I):** Go to the folder *part2/monitoringwithinterrupts* and complete the C code to increase the value of the variable *button_press* using an interrupt. Check that your code is working. When does the number of button presses get updated on the screen? Why? Does it correspond to the number of times you pressed the button? If not, can you explain why and can you fix your code to display exactly the number of times the button was pressed?

    Solution:
    Add this function

```
ISR(PCINT0_vect)
{
  button_press++;
}
```

    The number of button presses gets updated once every 10 seconds, but the correct number of presses (multiplied by two → see solution of question 9) is recorded. To only display the number of presses, you can modify the print to display half of the button presses:

```
oled.print("Button pressed "+String(button_press/2)+"
times");
```

12. **(Q):** Reflect on the lab so far. How could the code in part1 be improved with the knowledge you have now on interrupts?

    Solution: We could attach interrupts to all three buttons in the basestation code to ensure that all changes requested by the user are recorded. In reality this is not possible with this board because of some hardware constraints, but it would

theoretically be possible with a redesign of the hardware that allows all the buttons to be attached to an interrupt pin of the microcontroller. Right now only one of the buttons at a time can be configured as an interrupt.

Part 3: Energy-aware communication

13. **(I)**: Implement the scenario above by detecting when the temperature reaches a certain stage and changing the value of the frequency of data transmission accordingly. **Do not forget to change the PAN ID on line 44** to the number that you have used in previous questions. Upload your code on the remote node, then connect the base station to the computer and open the serial plotter. Test your implementation by placing a finger on the temperature sensor to increase its temperature.

    Solution:

```
if (t>28.0){
   loop_time=100;
 }else{
   loop_time=1000;
 }
```

IMPORTANT: What to do before handing the board back to the TAs

Before handing the boards back to a TA, go to the folder *maintenance* and upload the code you find inside on both boards. This code reads the battery voltage and prints it on the OLED screen. This way, we can quickly turn the boards on and see if they need to be recharged. Thank you for your help!