ASSIGNMENT 2

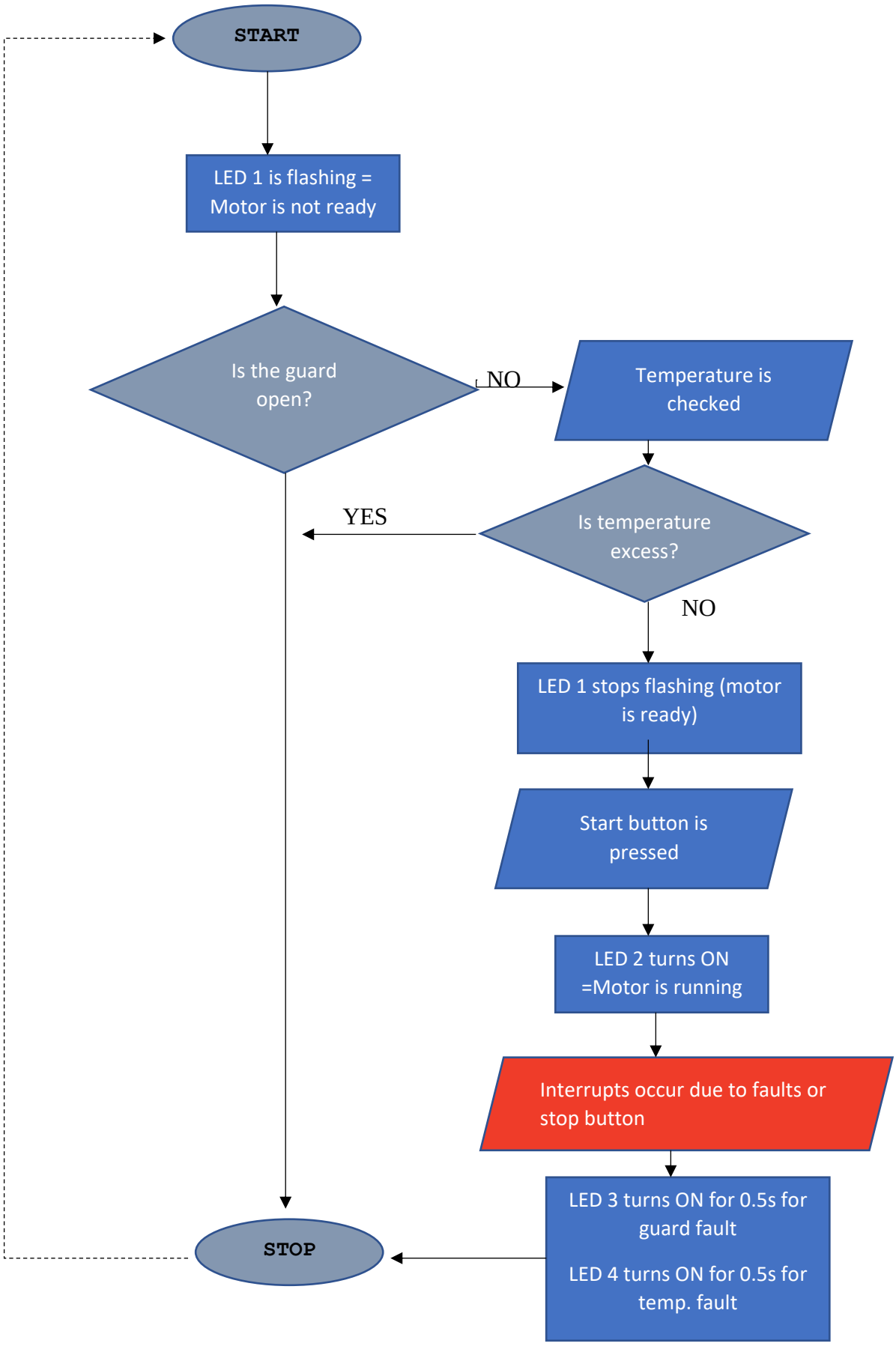
KIE4022: EMBEDDED SYSTEMS

Lecturer: **Dr. Mohamad Sofian Bin Abu Talip**

Group Members:

1. **Abdolraouf Rahmani (KIE170720 | 17080764)**
2. **Mohd Riduwan Bin Mohd Basir (KIE170059 | 17092735)**

PROGRAM STRUCTURE:



The Program

It consists of 2 structures:

1) Main structure
2) Sub-loop structure

Main Structure:

In this part, the outputs will be determined based on the inputs (conditions) of the motor that is defined. It has 2 major loops for the state of ready=0 and ready =1. These two loops will be further seen in the sub-loop structure. Aside from that, the variables for the interrupt inputs are called as a way to control the condition of the motor by using the buttons and switches.

Sub-loop structure:

i) Ready = 0

This state indicates that the motor is not yet ready to run. The LED 1 which is an output that would flash in the "not ready" state.

"Not ready" state is the indication that fault might occur during that stage. The fault might be due to the guard that is opened, or the temperature has exceeded the acceptable value.

This loop will keep running until all the parameters for the guard and temperature are at 0 state.

ii) Ready = 1

This state indicates that the motor is ready to run. The LED 1 will stop flashing. Once the start button is pressed, the LED 2 which is an output corresponds to the start button will start to turn ON. This means that the motor is currently running without any interruption. The other variables for the outputs are in 0 state except for the LED 2.

In the case where interrupts occur during the running stage, the LED 3 would turn ON for 0.5s follows by the flashing of LED 1. This is to show that guard is opened. For temperature fault, the LED 4 would turn ON for 0.5s instead of LED 3.

iii) Interrupts

After the occurrence of the interrupt, which is due to the fault, the program would exit from the loop of ready=1 and stop the motor from running. The program gets back initial state where all the conditions are ensured to be in the acceptable state before letting the motor to run.

SIMULATION CODE

```
1.  #include "mbed.h"
2.
3. #define DEBUG
4.
5. //inputs
6. #define START_BUTTON_1 p5
7. #define STOP_BUTTON_2 p6
8. #define GUARD_SWITCH_1 p7
9. #define TEMP_SWITCH_2 p8
10.
11.  //outputs
12.  #define LED1_MOTOR_READY   LED1
13.  #define LED2_MOTOR_RUNNIG  LED2
14.  #define LED3_GUARD_FAULT   LED3
15.  #define LED4_TMEP_FAULT    LED4
16.
17.
18.  //declaring input and output pins
19.  DigitalOut led_motor_ready(LED1_MOTOR_READY);
20.  DigitalOut led_motor_running(LED2_MOTOR_RUNNIG);
21.  DigitalOut led_temp_fault(LED4_TMEP_FAULT);
22.  DigitalOut led_guard_fault(LED3_GUARD_FAULT);
23.
24.  //tickers and timeouts
25.  Ticker flipper; //flashing
26.  Timeout temp_timeout;
27.  Timeout gaurd_timeout;
28.
29.  //Define Interrupt Inputs
30.  InterruptIn start(START_BUTTON_1);
31.  InterruptIn stop(STOP_BUTTON_2);
32.  InterruptIn guard(GUARD_SWITCH_1);
33.  InterruptIn temp(TEMP_SWITCH_2);
34.
35.  //global variables. Their value determines initial state when motor
   is powered.
36.  bool ready=              0; //not ready
37.  bool stoped=             0; //0 means stop button not being held
38.  bool running=            0; //motor not running
39.  bool guard_fault=        0; //guard fault hasnt occured. This is
   used to trigger fualt LED3.
40.  bool temp_fault=         0; //temp fault hasnt occured. This is
   used to trigger fualt LED4.
41.  bool guard_open=         0; //guard is closed
42.  bool temp_high=          0; //temperature is not excess
43.  bool MotorWasRunning=    0; //gives the status of the motor just
   moments before.
44.  bool startflashing=      0; //states whether to start flashing or
   not
45.
46.  //Define ISRs for interrupts
47.  void start_handler(){
48.          running =   1;
49.  }
50.
51.  void stoped_handler(){//when stop button is pressed
52.          ready =     0;
53.          running =   0;
```

```
54.          stoped=     1;
55.          #ifdef DEBUG
56.          printf("DEBUG:
   ready=%i,stoped=%i,running=%i,guard_fault=%i,temp_fault=%i,\n      :
   guard_open=%i,temp_high=%i,MotorWasRunning=%i,startflashing=%i\n",rea
   dy,stoped,running,guard_fault,temp_fault,guard_open,temp_high,MotorWa
   sRunning,startflashing);
57.      #endif
58.
59.  }
60.
61.  void not_stoped_handler(){ //when stop button is realeased
62.          stoped=     0;
63.  }
64.
65.  void temp_high_handler(){ //when temp is higher than preset value
66.          ready =     0;
67.          running =   0;
68.          temp_fault= 1;
69.          temp_high=  1;
70.  }
71.
72.  void temp_low_handler(){ //when temp is lower than preset value
73.          temp_high=0;
74.  }
75.
76.  void guard_open_handler(){ //when guard is opened
77.          ready =     0;
78.          running =   0;
79.          guard_fault= 1;
80.          guard_open=  1;
81.  }
82.
83.  void guard_close_handler(){ //when guard is closed
84.          guard_open=  0;
85.  }
86.
87.  void flip() {
88.      led_motor_ready = !led_motor_ready;
89.  }
90.
91.  void temp_fault_alert() {
92.      led_temp_fault = 0; //turn off after 0.5second
93.  }
94.
95.  void gaurd_fault_alert() {
96.      led_guard_fault = 0; //turn off after 0.5seconds
97.
98.  }
99.
100. // MAIN FUNCTION *************************
101. int main() {
102.  while(1){
103.     #ifdef DEBUG
104.     printf("                    START%s\n", "");
105.     #endif
106.     //Interrupt handlers
107.         start.rise(&start_handler);
108.
109.         stop.rise(&stoped_handler);
110.         stop.fall(¬_stoped_handler);
```

```
111.
112.         guard.rise(&guard_open_handler);
113.         guard.fall(&guard_close_handler);
114.
115.         temp.rise(&temp_high_handler);
116.         temp.fall(&temp_low_handler);
117.
118.     startflashing=1;
119.     led_motor_ready=0;
120.
121.     while (ready==0) {
122.         led_motor_running=0; //turn off the motor
123.
124.         if(startflashing==1){
125.             flipper.attach(&flip, 2.0); // the address of the
    function to be attached (flip) and the interval (2 seconds)
126.             startflashing=0;
127.             #ifdef DEBUG
128.             printf("NOT READY:  start flashing%s\n","");
129.             #endif
130.         }
131.
132.         if (temp_fault==1){
133.             led_temp_fault=1;
134.             temp_timeout.attach(&temp_fault_alert, 0.5); // timeout
    of 0.5s
135.             temp_fault=0;
136.             printf("FAULT:  TEMP%s\n","");
137.         }
138.
139.         if (guard_fault==1){
140.             if(MotorWasRunning){
141.                     led_guard_fault=1;
142.                     gaurd_timeout.attach(&gaurd_fault_alert, 0.5);
    // timeout of 0.5s
143.                     MotorWasRunning=0;
144.                     #ifdef DEBUG
145.                     printf("FAULT:  GUARD%s\n","");
146.                     #endif
147.             }
148.         guard_fault=0;
149.         }
150.
151.         if(guard_open==0 && temp_high==0 && stoped==0){
152.                 ready=1;
153.         //      printf("STATUS CHANGE: Ready Now%s\n", "");
154.         }
155.         wait_ms(200);
156.     }//end of not ready loop
157.
158.     #ifdef DEBUG
159.       bool debug1=1;
160.       bool debug2=1;
161.         #endif
162.
163.     while(ready==1){//ready loop
164.
165.         #ifdef DEBUG
166.         if(debug1==1){
167.             printf("READY%s\n", "");
168.             debug1=0;
```

```
169.         }
170.         #endif
171.
172.             led_temp_fault=0;
173.         led_guard_fault=0;
174.             led_motor_ready=1;
175.             flipper.detach();
176.
177.             while(running==1 && ready==1){
178.                 #ifdef DEBUG
179.                 if(debug2==1){
180.                     printf("RUNNING%s\n", "");
181.                     debug2=0;
182.                 }
183.                 #endif
184.                     led_motor_running=1;
185.                     led_motor_ready=0;
186.                     MotorWasRunning=1;
187.                 wait_ms(200);
188.             }
189.             wait_ms(200);
190.     }
191.     #ifdef DEBUG
192.     printf("                    END%s\n", "");
193.     #endif
194. }//full while loop
195. }
```

SIMULATION

In the figure 1, the program is in the right condition (no fault).

LED 1 does not flash indicating that the motor is ready to run.

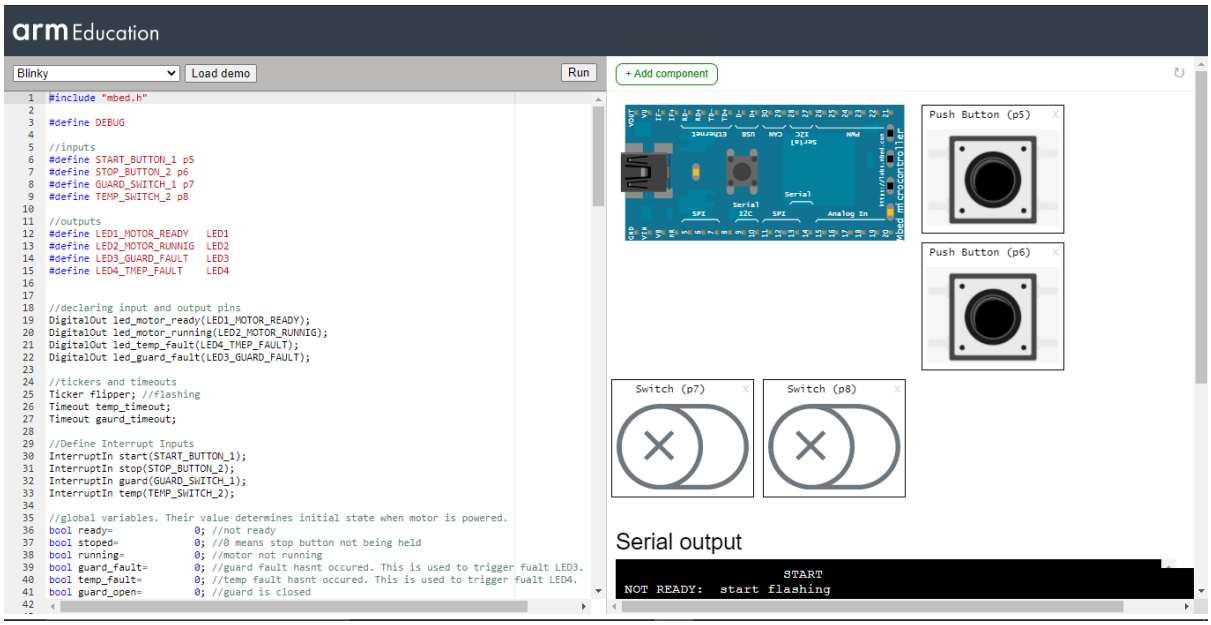


Figure 1

In Figure 2, the start button is pressed, and the LED 2 turns ON. This is to show that the motor is running.

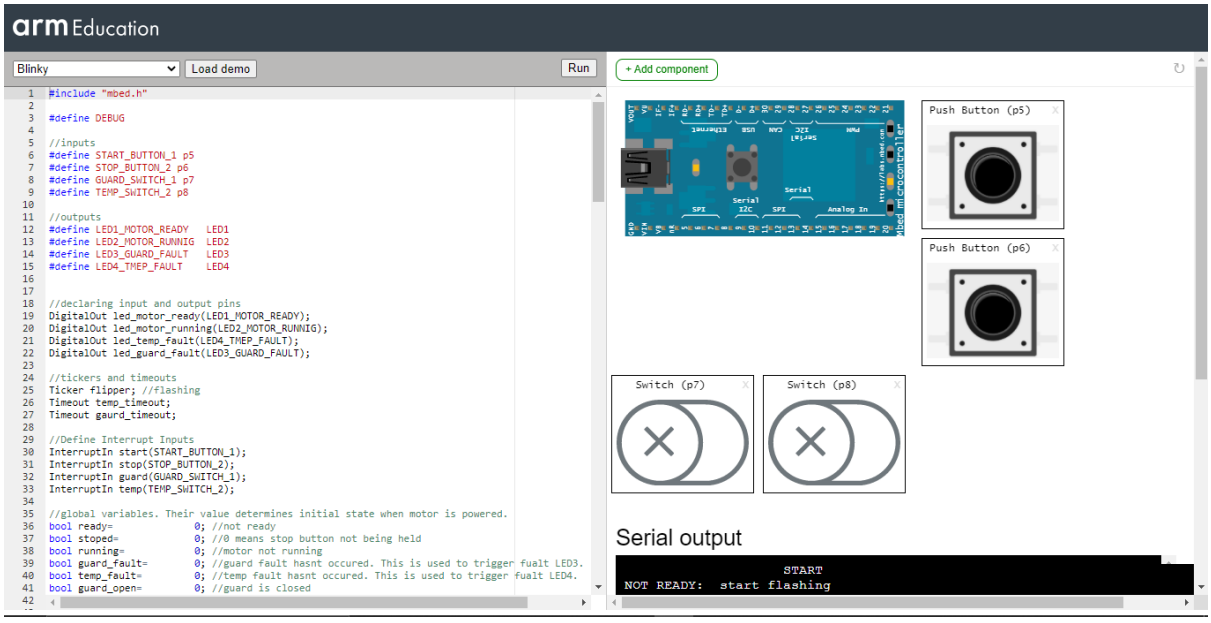


Figure 2

In Figure 3, the guard is opened which indicated by the switch p7. LED 3 turns ON for 0.5s before the LED 1 starts flashing.

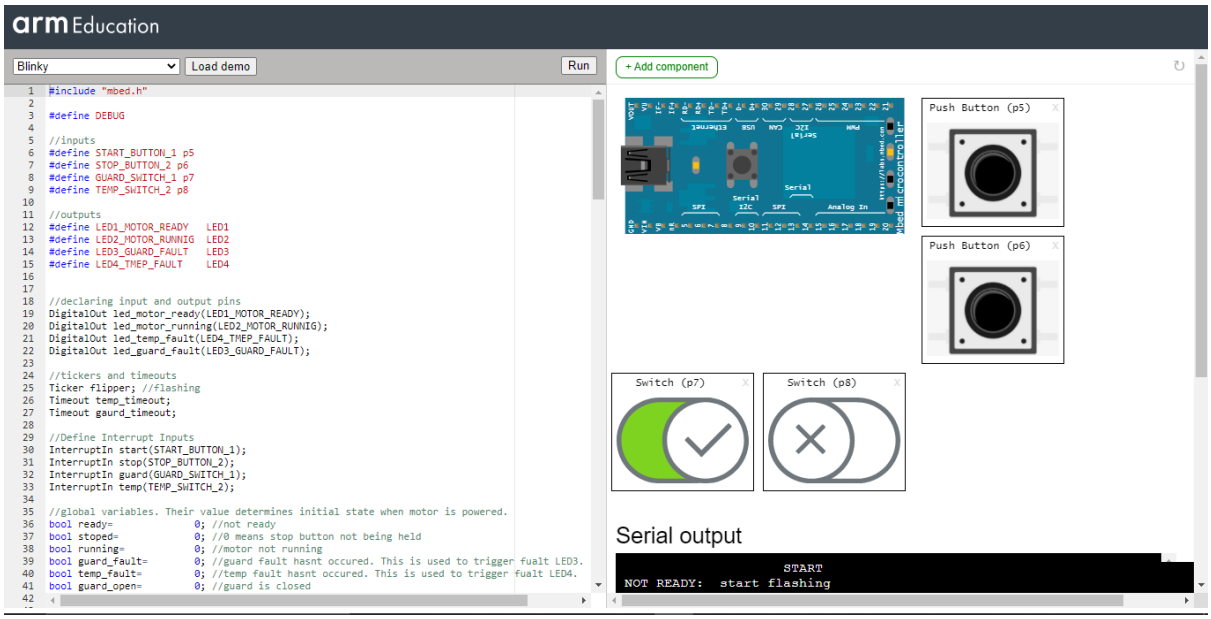


Figure 3

In Figure 4, the temperature is exceeded the acceptable value which indicated by the switch p8. LED 4 turns ON for 0.5s before the LED 1 starts flashing.

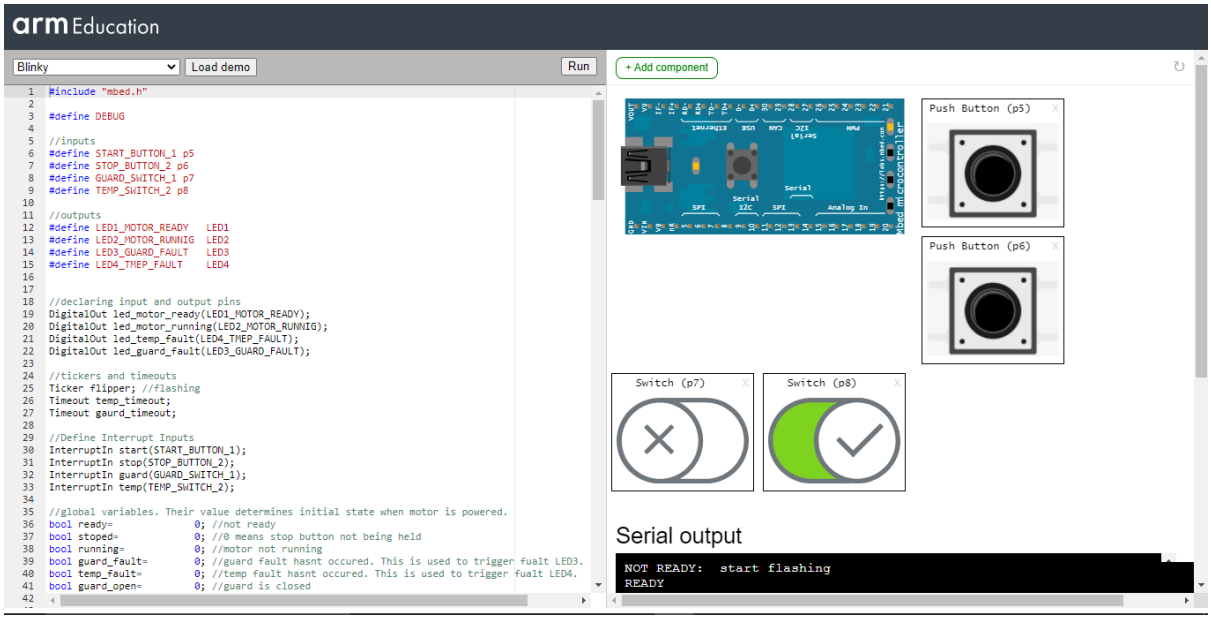


Figure 4

In Figure 5, both faults occur. Thus, the LED 1 starts flashing.

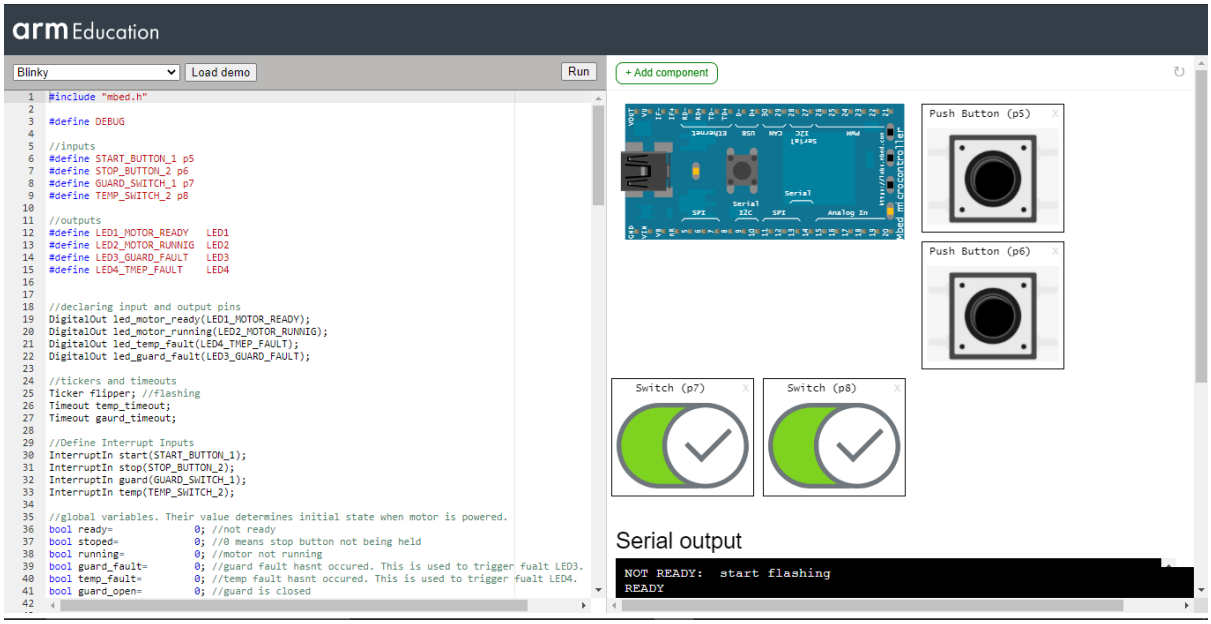


Figure 5