

[SMART_FARM]

FINAL REPORT

학번 : 2013075011

성명 : 박 성 현

교수 : 이 은 지

목차

1. 최종 프로젝트

1.1 요구사항

1.2 알고리즘 설계 및 개념

1.3 소스코드

1.4 고찰

1.5 참고 자료

2. 쓰레드 프로젝트

2.1 요구사항

2.2 알고리즘 설계 및 개념

2.3 소스코드

2.4 고찰

2.5 참고 자료

1. 프로젝트 명 : SMART_FARM FINAL

1.1. 요구사항

1	밀리초마다 온도와 조도를 모니터링 하라
2	조도값을 측정할 땐 아날로그 입력을 이용하라
3	서버를 만들고, MySQL과 Apache를 설치하라
4	10초마다 서버로 센서데이터를 보내라
5	20도가 넘어가면 5초 동안 선풍기를 틀어라
6	특정 조도를 넘기면 LED가 작동하도록 하라
7	각 기능들이 독립적으로 작동하도록 하라(4개의 쓰레드를 사용하라)
8	쓰레드끼리 시그널을 통해 통신하도록 하라
9	1분 이상 데이터를 수집하라
10	웹 서버의 데이터 모습을 캡처하라 (이 화면이 GIT과 최종보고서에 포함되어야한다)

1.2. 알고리즘 설계 및 개념

MAIN
데이터 베이스, wiringPI, signal 핸들러, PINMODE를 초기화한다 각 기능의 쓰레드를 생성한다 pthread_t producer, p 각 쓰레드를 시작한다.
PRODUCER
WHILE(1) LOCK을 건다 WHILE (COUNT == MAX) WAIT한다, EMPTY신호가 올 때까지 온도 센서 값과 조도 센서값을 읽고 PUT(), 공유버퍼에 값을 넣고, COUNT를 증가한다. LOCK을 푼다
DATA_SENDER
WHILE(1) LOCK을 건다 WHILE (COUNT == 0) WAIT한다, FILL신호가 올 때까지 GET(), 공유버퍼에서 값을 빼고, COUNT를 감소한다. 쿼리를 통해 서버로 값을 보낸다.\n LOCK을 푼다 10초간 대기
FAN_CONTROLLER
WHILE(1) LOCK을 건다 WHILE (COUNT == 0) WAIT한다, FILL신호가 올 때까지 IF 온도 > 20.0f digitalWrite(FAN, 1) LOCK을 푼다 5초간 대기 digitalWrite(FAN, 0)
LED_CONTROLLER
WHILE(1) LOCK을 건다 WHILE (COUNT == 0) WAIT한다, FILL신호가 올 때까지 GET(), 공유버퍼에서 값을 빼고, COUNT를 감소한다. IF 조도 값이 2500 이상 digitalWrite(LED, 1) ELSE digitalWrite(LED, 0) LOCK을 푼다

1.3. 소스 코드

```
1  #include <stdio.h>
2  #include <pthread.h>
3  #include <stdlib.h>
4  #include <unistd.h>
5  #include <mysql/mysql.h>
6  #include <sys/types.h>
7  #include <errno.h>
8  #include <wiringPi.h>
9  #include <stdint.h>
10 #include <string.h>
11 #include <signal.h>
12 #include <softPwm.h>
13 #include <wiringPiSPI.h>
14 #include <time.h>
15
16
17 #define DBHOST "18.222.59.249"
18 #define DBUSER "root"
19 #define DBPASS "root"
20 #define DBNAME "project_farm"
21
22 #define CS_MCP3208 11
23 #define SPI_CHANNEL 0
24 #define SPI_SPEED 1000000
25 #define MAX 1
26 #define MAXTIMINGS 85
27 #define RED 7
28 #define GREEN 9
29 #define BLUE 8
30 #define FAN 22
31
32 static int DHTPIN = 11;
33 static int dht22_dat[5] = {0,0,0,0,0};
34 static uint8_t sizecvt(const int read);
35
36 MYSQL *connector;
37 MYSQL_RES *result;
38 MYSQL_ROW row;
39
40 static int read_dht22_dat();
41 char query[1024];
42
43 float temp_buffer[MAX];
44 int light_buffer[MAX];
45
46 int fill_ptr = 0;
47 int use_ptr = 0;
48 int count = 0;
49
50 float temperature = 0.0f;
51 unsigned char adcChannel_light = 0;
52 int adcValue_light = 0;
53 float vout_light;
```

```

54 float vout_oftemp;
55 float percentrh = 0;
56 float supsiondo = 0;
57
58 pthread_cond_t empty, fill;
59 pthread_mutex_t mutex;
60
61 void get();
62 void put(float tmp, int lht);
63
64 void timer_for_data();
65 void *getSensorData(void* arg);
66 void *sendSensorData(void* arg);
67 void *fan_controller(void* arg);
68 void *LED_controller(void* arg);
69 void sig_handler(int signo); // 마지막 종료 함수
70 int read_mcp3208_adc(unsigned char adcChannel);
71
72 int main(int argc, char* argv[]){
73     signal(SIGINT, (void *)sig_handler);
74     if (wiringPiSetup () == -1)
75     {
76         fprintf(stdout, "Unable to start wiringPi: %s\n", strerror(errno));
77         return 1 ;
78     }
79
80     connector = mysql_init(NULL);
81     if (!mysql_real_connect(connector, DBHOST, DBUSER, DBPASS, DBNAME, 3306, NULL, 0))
82     {
83         fprintf(stderr, "%s\n", mysql_error(connector));
84         return 0;
85     }
86     if(wiringPiSPISetup(SPI_CHANNEL, SPI_SPEED) == -1)
87     {
88         fprintf(stdout, "wiringPiSPISetup Failed :%s\n", strerror(errno));
89         return 1;
90     }
91
92     pinMode(CS_MCP3208, OUTPUT);
93     pinMode(RED, OUTPUT);
94     pinMode(GREEN, OUTPUT);
95     pinMode(BLUE, OUTPUT);
96     pinMode(FAN, OUTPUT) ;
97
98     pthread_t producer, send2server, fan_manager, light_manager;
99     pthread_create(&producer, NULL, getSensorData, "prod");
100    pthread_create(&send2server, NULL, sendSensorData, "s2s");
101    pthread_create(&fan_manager, NULL, fan_controller, "fm");
102    pthread_create(&light_manager, NULL, LED_controller, "lm");
103
104    pthread_join(producer, NULL);
105    pthread_join(send2server, NULL);
106    pthread_join(fan_manager, NULL);
107    pthread_join(light_manager, NULL);
108    return 0;

```

```

109 }
110
111 void *getSensorData(void* arg){
112     struct timespec delay_time2Produce;
113     while(1){
114         pthread_mutex_lock(&mutex);
115         while(count == MAX)
116             pthread_cond_wait(&empty, &mutex);
117         if(read_dht22_dat()){
118             put(temperature, adcValue_light);
119         }
120         delay_time2Produce.tv_sec = 0;
121         delay_time2Produce.tv_nsec = 1000;
122         pthread_cond_signal(&fill);
123         pthread_mutex_unlock(&mutex);
124         nanosleep(&delay_time2Produce, NULL);
125     }
126 }
127
128 void *sendSensorData(void* arg){
129     struct timespec delay_time2SendServer;
130     while(1){
131         pthread_mutex_lock(&mutex);
132         while(count==0)
133             pthread_cond_wait(&fill, &mutex);
134         get();
135         timer_for_data();
136         delay_time2SendServer.tv_sec = 10;
137         delay_time2SendServer.tv_nsec = 0;
138         pthread_cond_signal(&empty);
139         pthread_mutex_unlock(&mutex);
140         nanosleep(&delay_time2SendServer, NULL);
141     }
142 }
143
144 void *fan_controller(void* arg){
145     struct timespec delay_time2FAN_ON;
146     while(1){
147         pthread_mutex_lock(&mutex);
148         while(count==0)
149             pthread_cond_wait(&fill, &mutex);
150         get();
151         if(temperature > 20.0f)
152         {
153             digitalWrite(FAN, 1);
154             delay_time2FAN_ON.tv_sec = 5;
155             delay_time2FAN_ON.tv_nsec = 0;
156         }
157
158         pthread_cond_signal(&empty);
159         pthread_mutex_unlock(&mutex);
160         nanosleep(&delay_time2FAN_ON, NULL);
161         digitalWrite(FAN, 0);
162         delay_time2FAN_ON.tv_sec = 1;
163         nanosleep(&delay_time2FAN_ON, NULL);

```

```

164     }
165 }
166
167 void *LED_controller(void* arg)
168 {
169     while(1){
170         pthread_mutex_lock(&mutex);
171         while(count==0)
172             pthread_cond_wait(&fill, &mutex);
173         get();
174         if(adcValue_light > 2500)
175         {
176             digitalWrite(RED, 1);
177         }
178         else
179             digitalWrite(RED, 0);
180         pthread_cond_signal(&empty);
181         pthread_mutex_unlock(&mutex);
182     }
183 }
184
185 void timer_for_data(){
186     sprintf(query,"insert into farm_data values (now(),%f,%d)", temperature,adcValue_light);
187     mysql_query(connector, query);
188     printf("Uploaded to Server\n");
189 }
190
191 void put(float tmp, int lht){
192     temp_buffer[fill_ptr] = tmp;
193     light_buffer[fill_ptr] = lht;
194     fill_ptr = (fill_ptr+1)%MAX;
195     count++;
196 }
197
198 void get()
199 {
200     float tmp = temp_buffer[use_ptr];
201     int lht = light_buffer[use_ptr];
202     use_ptr = (use_ptr +1)%MAX;
203     count --;
204 }
205
206
207 static uint8_t sizecvt(const int read)
208 {
209     /* digitalWrite() and friends from wiringpi are defined as returning a value
210     < 256. However, they are returned as int() types. This is a safety function */
211
212     if (read > 255 || read < 0)
213     {
214         printf("Invalid data from wiringPi library\n");
215         exit(EXIT_FAILURE);
216     }
217     return (uint8_t)read;
218 }

```



```

219
220 static int read_dht22_dat()
221 {
222     uint8_t laststate = HIGH;
223     uint8_t counter = 0;
224     uint8_t j = 0, i;
225
226     dht22_dat[0] = dht22_dat[1] = dht22_dat[2] = dht22_dat[3] = dht22_dat[4] = 0;
227
228     // pull pin down for 18 milliseconds
229     pinMode(DHTPIN, OUTPUT);
230     digitalWrite(DHTPIN, HIGH);
231     delay(10);
232     digitalWrite(DHTPIN, LOW);
233     delay(18);
234     // then pull it up for 40 microseconds
235     digitalWrite(DHTPIN, HIGH);
236     delayMicroseconds(40);
237     // prepare to read the pin
238     pinMode(DHTPIN, INPUT);
239
240     // detect change and read data
241     for ( i=0; i< MAXTIMINGS; i++) {
242         counter = 0;
243         while (sizecvt(digitalRead(DHTPIN)) == laststate) {
244             counter++;
245             delayMicroseconds(1);
246             if (counter == 255) {
247                 break;
248             }
249         }
250         laststate = sizecvt(digitalRead(DHTPIN));
251
252         if (counter == 255) break;
253
254         // ignore first 3 transitions
255         if ((i >= 4) && (i%2 == 0)) {
256             // shove each bit into the storage bytes
257             dht22_dat[j/8] <<= 1;
258             if (counter > 50)
259                 dht22_dat[j/8] |= 1;
260             j++;
261         }
262     }
263
264     // check we read 40 bits (8bit x 5 ) + verify checksum in the last byte
265     // print it out if data is good
266     if ((j >= 40) &&
267         (dht22_dat[4] == ((dht22_dat[0] + dht22_dat[1] + dht22_dat[2] + dht22_dat[3]) & 0xFF)) ) {
268         float t, h;
269         int tint =0;
270         h = (float)dht22_dat[0] * 256 + (float)dht22_dat[1];
271         h /= 10;
272         t = (float)(dht22_dat[2] & 0x7F)* 256 + (float)dht22_dat[3];
273         t /= 10.0;

```

```

274         if ((dht22_dat[2] & 0x80) != 0) t *= -1;
275         tint = t;
276         adcValue_light = read_mcp3208_adc(adcChannel_light);
277         printf("Temperature = %.2f *C ,light sensor = %d\n", t, adcValue_light );
278         temperature = t;
279         return 1;
280     }
281     else
282     {
283         //printf("Data not good, skip\n");
284         return 0;
285     }
286 }
287
288 int read_mcp3208_adc(unsigned char adcChannel)
289 {
290     unsigned char buff[3];
291     int adcValue = 0;
292
293     buff[0] = 0x06 | ((adcChannel & 0x07) >> 2);
294     buff[1] = ((adcChannel & 0x07) << 6);
295     buff[2] = 0x00;
296
297     digitalWrite(CS_MCP3208, 0);
298     wiringPiSPIDataRW(SPI_CHANNEL, buff, 3);
299
300     buff[1] = 0x0f & buff[1];
301     adcValue = (buff[1] << 8 ) | buff[2];
302
303     digitalWrite(CS_MCP3208, 1);
304
305     return adcValue;
306 }
307
308
309 void sig_handler(int signo)
310 {
311     printf("process stop\n");
312     digitalWrite(RED, 0);
313     digitalWrite(GREEN, 0);
314     digitalWrite(BLUE, 0);
315     digitalWrite (FAN, 0) ;
316     exit(0);
317 }

```

1.4. 고찰

사실 한 프로세스의 단위를 스레드로 나누는 것이, 실시간 처리에 가깝도록, 각 센서들이 병행하게 동작하도록 하려는 것이지만, delay함수와 sleep함수는 생각보다 체감하는 지연시간이 크다. 그래서 제일 좋은 방법은 처음엔 타이머 인터럽트라고 생각했다. 타이머 인터럽트를 이용해서, 10초마다 서버로 보내고, 5초 동안만 키고 끄도록 하려고 조사한 결과, timer와 관련된 리눅스의 함수가 있었다. timer_create와 set_itimer같은 함수들을 통해 타이머를 생성하고, 실행할 수 있었으나, 동적으로 생성되었다가, 타이머를 종료할 수가 없었다. 즉, 타이머 인터럽트를 20도가 높을 때에만 타이머를 등록하고, 5초 뒤에 소멸하는 것은 생각보다 너무 어려웠다. 따라서, nanosleep을 이용해서 나노초 단위로 쓰레드 자체가 시간을 재도록 만들었다. sleep이나 delay보다는 REAL_TIME에 더 충실하다고 생각한다. 쓰레드의 스케줄링에 대해서도 고려해보며 환경큐가 왜 쓰였는지 알게 되었다.

1.5. 참고 자료

아마존 웹 서버(AWS)에서 인스턴스를 생성한 화면

The screenshot shows the AWS Management Console interface. On the left, the navigation menu includes 'EC2 대시보드', '이벤트', '태그', '보고서', '제한', '인스턴스', '인스턴스 시작 템플릿', and '시작 템플릿'. The main content area displays a table of EC2 instances. One instance is listed with ID 'i-0f62af59d189d9a5a', type 't2.micro', region 'us-east-2c', and status 'running'.

Name	인스턴스 ID	인스턴스 유형	가용 영역	인스턴스 상태
	i-0f62af59d189d9a5a	t2.micro	us-east-2c	running

phpmyadmin에서 데이터베이스를 확인한 화면

The screenshot shows the phpMyAdmin interface. The left sidebar displays a tree view of databases and tables, including 'project_farm' and 'farm_data'. The main content area shows the 'farm_data' table with columns 'time', 'temperature', and 'lightness'. The table contains 10 rows of data, each with a timestamp, temperature, and lightness value.

	time	temperature	lightness
<input type="checkbox"/> 수정 <input type="checkbox"/> 복사 <input type="checkbox"/> 삭제	2018-05-23 02:38:49	27.8	1886
<input type="checkbox"/> 수정 <input type="checkbox"/> 복사 <input type="checkbox"/> 삭제	2018-05-23 02:39:04	27.9	2128
<input type="checkbox"/> 수정 <input type="checkbox"/> 복사 <input type="checkbox"/> 삭제	2018-05-23 02:39:15	27.9	2785
<input type="checkbox"/> 수정 <input type="checkbox"/> 복사 <input type="checkbox"/> 삭제	2018-05-23 02:39:25	27.9	2871
<input type="checkbox"/> 수정 <input type="checkbox"/> 복사 <input type="checkbox"/> 삭제	2018-05-23 02:39:35	28	2105
<input type="checkbox"/> 수정 <input type="checkbox"/> 복사 <input type="checkbox"/> 삭제	2018-05-23 02:39:46	28	2823
<input type="checkbox"/> 수정 <input type="checkbox"/> 복사 <input type="checkbox"/> 삭제	2018-05-23 02:39:56	28	1743
<input type="checkbox"/> 수정 <input type="checkbox"/> 복사 <input type="checkbox"/> 삭제	2018-05-23 02:40:06	28	1534
<input type="checkbox"/> 수정 <input type="checkbox"/> 복사 <input type="checkbox"/> 삭제	2018-05-23 02:40:16	28	1640

서버 내의 php파일에서 데이터 통계를 확인하는 화면

IE에서 가져온 북마크 내 블로그 OLC CENTER Apache-Tomcat 외부 자을리

Connect Success!학번: 2013075011성명: 박성현

날짜 및 시간	온도	조도
2018-05-23 02:40:16	28	1640
2018-05-23 02:40:06	28	1534
2018-05-23 02:39:56	28	1743
2018-05-23 02:39:46	28	2823
2018-05-23 02:39:35	28	2105
2018-05-23 02:39:25	27.9	2871
2018-05-23 02:39:15	27.9	2785
2018-05-23 02:39:04	27.9	2128
2018-05-23 02:38:49	27.8	1886

1ms마다 생산자가 값을 저장하고, 모니터링하는 모습(중간에 서버 전송하면 문구를 출력)

```
pi@raspberrypi: ~  
File Edit Tabs Help  
Temperature = 28.00 *C ,light sensor = 1647  
Temperature = 28.00 *C ,light sensor = 1636  
Temperature = 28.00 *C ,light sensor = 1626  
Temperature = 28.00 *C ,light sensor = 1642  
Temperature = 28.00 *C ,light sensor = 1648  
Temperature = 28.00 *C ,light sensor = 1645  
Temperature = 28.00 *C ,light sensor = 1637  
Temperature = 28.00 *C ,light sensor = 1624  
Temperature = 28.00 *C ,light sensor = 1638  
Temperature = 28.00 *C ,light sensor = 1647  
Temperature = 28.00 *C ,light sensor = 1645  
Temperature = 28.00 *C ,light sensor = 1640  
Temperature = 28.00 *C ,light sensor = 1630  
Temperature = 28.00 *C ,light sensor = 1640  
Uploaded to Server  
Temperature = 28.00 *C ,light sensor = 1644  
Temperature = 28.00 *C ,light sensor = 1635  
Temperature = 28.00 *C ,light sensor = 1619  
Temperature = 28.00 *C ,light sensor = 1634  
Temperature = 28.00 *C ,light sensor = 1635  
Temperature = 28.00 *C ,light sensor = 1612  
Temperature = 28.00 *C ,light sensor = 1596  
Temperature = 28.00 *C ,light sensor = 1589  
Temperature = 28.00 *C ,light sensor = 1609
```

2. 프로젝트 명 : THREAD_PROGRAMMING_PRACTICE

2.1. 요구사항

1	생산자는 온도 값을 3초마다 측정한다.
2	생산자는 측정한 온도 값을 공유버퍼에 저장한다.
3	생산자는 시그널링을 통해 소비자를 깨운다.
4	소비자는 데이터베이스로 온도값을 저장한다.

2.2. 알고리즘 설계 및 개념

MAIN
데이터 베이스, wiringPI, signal 핸들러, PINMODE를 초기화한다 각 기능의 쓰레드를 생성한다 pthread_t producer, p 각 쓰레드를 시작한다.
PRODUCER
WHILE(1) LOCK을 건다 WHILE (COUNT == MAX) WAIT한다, EMPTY신호가 올 때까지 온도 센서 값을 읽고 PUT(), 공유버퍼에 값을 넣고, COUNT를 증가한다. LOCK을 푼다 3초간 대기한다
DATA_SENDER
WHILE(1) LOCK을 건다 WHILE (COUNT == 0) WAIT한다, FILL신호가 올 때까지 GET(), 공유버퍼에서 값을 빼고, COUNT를 감소한다. 쿼리를 통해 데이터베이스에 값을 저장한다. LOCK을 푼다

2.3. 소스 코드

```
1  #include <errno.h>
2  #include <wiringPi.h>
3  #include <stdint.h>
4  #include <string.h>
5  #include <signal.h>
6  #include <softPwm.h>
7
8  #define MAX 5
9  #define MAXTIMINGS 85
10 #define DBHOST "127.0.0.1"
11 #define DBUSER "root"
12 #define DBPASS "root"
13 #define DBNAME "demofarmdb"
14
15 static int DHTPIN = 11;
16 static int dht22_dat[5] = {0,0,0,0,0};
17 static uint8_t sizecvt(const int read);
18
19 MYSQL *connector;
```

```

20  MYSQL_RES *result;
21  MYSQL_ROW row;
22
23  static int read_dht22_dat();
24  char query[1024];
25
26  int loops = 1000;
27  float buffer[MAX];
28  int fill_ptr = 0;
29  int use_ptr = 0;
30  int count = 0;
31  float temperature = 0.0f;
32
33  pthread_cond_t empty, fill;
34  pthread_mutex_t mutex;
35
36  float get();
37  void put(float value);
38  void *consumer(void* arg);
39  void *producer(void* arg);
40
41  int main(int argc, char* argv[]){
42      if (wiringPiSetup () == -1)
43      {
44          fprintf(stdout, "Unable to start wiringPi: %s\n", strerror(errno));
45          return 1 ;
46      }
47
48      connector = mysql_init(NULL);
49      if (!mysql_real_connect(connector, DBHOST, DBUSER, DBPASS, DBNAME, 3306, NULL, 0))
50      {
51          fprintf(stderr, "%s\n", mysql_error(connector));
52          return 0;
53      }
54
55      pthread_t prod, cons1, cons2;
56      pthread_create(&prod, NULL, producer, "prod");
57      pthread_create(&cons1, NULL, consumer, "cons1");
58      pthread_create(&cons2, NULL, consumer, "cons2");
59
60      pthread_join(prod, NULL);
61      pthread_join(cons1, NULL);
62      pthread_join(cons2, NULL);
63      return 0;
64  }
65
66  void *producer(void* arg){
67      int i;
68      while(1){
69          pthread_mutex_lock(&mutex);
70          while(count == MAX)
71              pthread_cond_wait(&empty, &mutex);
72          if(read_dht22_dat()){
73              put(temperature);
74              printf("%s putting %.2f\n", (void*)arg, temperature);

```

```

75     }
76     pthread_cond_signal(&fill);
77     pthread_mutex_unlock(&mutex);
78     sleep(3);
79 }
80 }
81 void *consumer(void* arg){
82     int i;
83     while(1){
84         pthread_mutex_lock(&mutex);
85         while(count==0)
86             pthread_cond_wait(&fill, &mutex);
87         float tmp = get();
88         mysql_query(connector, query);
89         printf("%s consume %.2f\n", (void*)arg, tmp);
90         pthread_cond_signal(&empty);
91         pthread_mutex_unlock(&mutex);
92     }
93 }
94
95 void put(float value){
96     buffer[fill_ptr] = value;
97     fill_ptr = (fill_ptr+1)%MAX;
98     count++;
99 }
100
101 float get()
102 {
103     float tmp = buffer[use_ptr];
104     use_ptr = (use_ptr + 1)%MAX;
105     count --;
106     return tmp;
107 }
108
109 static uint8_t sizecvt(const int read)
110 {
111     if (read > 255 || read < 0)
112     {
113         printf("Invalid data from wiringPi library\n");
114         exit(EXIT_FAILURE);
115     }
116     return (uint8_t)read;
117 }
118
119 static int read_dht22_dat()
120 {
121     uint8_t laststate = HIGH;
122     uint8_t counter = 0;
123     uint8_t j = 0, i;
124     dht22_dat[0] = dht22_dat[1] = dht22_dat[2] = dht22_dat[3] = dht22_dat[4] = 0;
125     pinMode(DHTPIN, OUTPUT);
126     digitalWrite(DHTPIN, HIGH);
127     delay(10);
128     digitalWrite(DHTPIN, LOW);
129     delay(18);

```

```

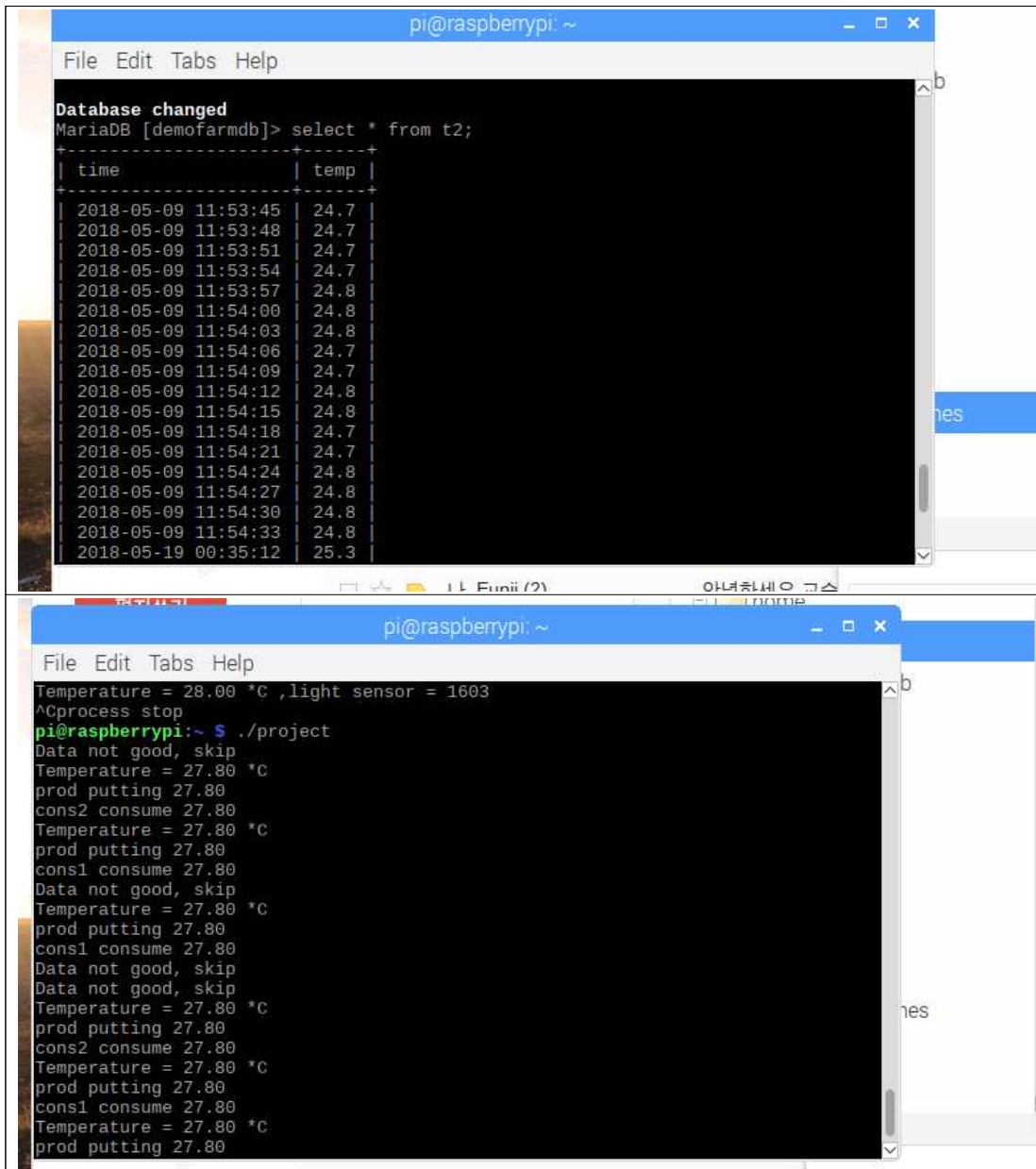
130    digitalWrite(DHTPIN, HIGH);
131    delayMicroseconds(40);
132    pinMode(DHTPIN, INPUT);
133    for ( i=0; i< MAXTIMINGS; i++) {
134        counter = 0;
135        while (sizecvt(digitalRead(DHTPIN)) == laststate) {
136            counter++;
137            delayMicroseconds(1);
138            if (counter == 255) {
139                break;
140            }
141        }
142        laststate = sizecvt(digitalRead(DHTPIN));
143        if (counter == 255) break;
144        if ((i >= 4) && (i%2 == 0)) {
145            dht22_dat[j/8] <=<= 1;
146            if (counter > 50)
147                dht22_dat[j/8] |= 1;
148            j++;
149        }
150    }
151    if ((j >= 40) &&
152        (dht22_dat[4] == ((dht22_dat[0] + dht22_dat[1] + dht22_dat[2] + dht22_dat[3]) & 0xFF)) ) {
153        float t, h;
154        int tint =0;
155        h = (float)dht22_dat[0] * 256 + (float)dht22_dat[1];
156        h /= 10;
157        t = (float)(dht22_dat[2] & 0x7F)* 256 + (float)dht22_dat[3];
158        t /= 10.0;
159        if ((dht22_dat[2] & 0x80) != 0) t *= -1;
160        tint = t;
161        sprintf(query,"insert into t2 values (now(),%f)", t);
162        printf("Temperature = %.2f *C \n", t );
163        temperature = t;
164        return 1;
165    }
166    else
167    {
168        printf("Data not good, skip\n");
169        return 0;
170    }
171 }

```

2.4. 고찰

교수님께서 주신 참고자료의 생산 소비자 코드를 참고하고, 교재의 mysql사용 코드를 참고했습니다. 온도센서관련 코드는 기존 프로젝트의 코드를 사용했습니다. 공유변수에서 값을 꺼내가는 순서가 따로 정해져 있지 않습니다. 따라서 누가 먼저 공유 버퍼의 값을 취할지 정하기 위해서는 일종의 스케줄링 기법을 적용해야 한다고 생각합니다. 교수님께서 IF 대신 WHILE문을 사용해야 생산자가 일을 마쳤을 때, 다른 쓰레드가 값을 가로채지 않는다는 강의가 생각났습니다.

2.5. 참고 자료



The image shows two screenshots of a terminal window on a Raspberry Pi, titled 'pi@raspberrypi: ~'. The window has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'.

The top screenshot shows the output of a SQL query in MariaDB. The prompt is 'MariaDB [demofarmdb]> select * from t2;'. The output is a table with two columns: 'time' and 'temp'. The data shows temperature readings from May 9, 2018, at 11:53:45 to 11:54:33, and one reading from May 19, 2018, at 00:35:12.

time	temp
2018-05-09 11:53:45	24.7
2018-05-09 11:53:48	24.7
2018-05-09 11:53:51	24.7
2018-05-09 11:53:54	24.7
2018-05-09 11:53:57	24.8
2018-05-09 11:54:00	24.8
2018-05-09 11:54:03	24.8
2018-05-09 11:54:06	24.7
2018-05-09 11:54:09	24.7
2018-05-09 11:54:12	24.8
2018-05-09 11:54:15	24.8
2018-05-09 11:54:18	24.7
2018-05-09 11:54:21	24.7
2018-05-09 11:54:24	24.8
2018-05-09 11:54:27	24.8
2018-05-09 11:54:30	24.8
2018-05-09 11:54:33	24.8
2018-05-19 00:35:12	25.3

The bottom screenshot shows the output of a command in the terminal. The prompt is 'pi@raspberrypi:~\$./project'. The output shows a series of messages related to a project, including temperature readings and sensor data. The messages are: 'Temperature = 28.00 *C ,light sensor = 1603', '^Cprocess stop', 'Data not good, skip', 'Temperature = 27.80 *C', 'prod putting 27.80', 'cons2 consume 27.80', 'Temperature = 27.80 *C', 'prod putting 27.80', 'cons1 consume 27.80', 'Data not good, skip', 'Temperature = 27.80 *C', 'prod putting 27.80', 'cons1 consume 27.80', 'Data not good, skip', 'Data not good, skip', 'Temperature = 27.80 *C', 'prod putting 27.80', 'cons2 consume 27.80', 'Temperature = 27.80 *C', 'prod putting 27.80', 'cons1 consume 27.80', 'Temperature = 27.80 *C', 'prod putting 27.80'.