1. **Titlu proiectului**

IoT - Client

2. **Nume student/studenti:**

Iorga Beatrice - 1407A

Hojbotă Cătălin-Constantin - 1407A

Ciumpoi Cătălin - 1407A

3. **Poze 3x4 cm**

4. **E_mail:**

beatrice.iorga@student.tuiasi.ro

catalin-constantin.hojbota@student.tuiasi.ro

catalin.ciumpoi@student.tuiasi.ro

5. **Rezumat**

Este creat un server prin XAMPP, micro-controller-ul fiind conectat la reteaua locala prin cablu ethernet. Serverul ruleaza constant pana la aparitia unui eveniment, acesta din urma fiind apasarea unuia din cele doua butoane. In urma evenimentului aplicatia trimite un email destinatarului prestabilit, in care avertizeaza modificarea starii micro-controller-ului. Dupa trimiterea mail-ului aplicatia revine la starea initiala, aceasta fiind numarator-ul pe biti. De asemenea in momentul in care este apasat un buton, este si afisat intr-un tabel evenimentul respectiv.

6. **Descriere resurse hardware utilizate (ARM)**

- XMC4800 Relax EtherCat Kit
- 1x cablu micro-USB
- 1x cablu Ethernet RJ45

7. **Descriere resurse software utilizate (DAVE App, Arduino, Mbed)**

- DAVE 4.4.2
- XAMPP WebServer 8.0.0

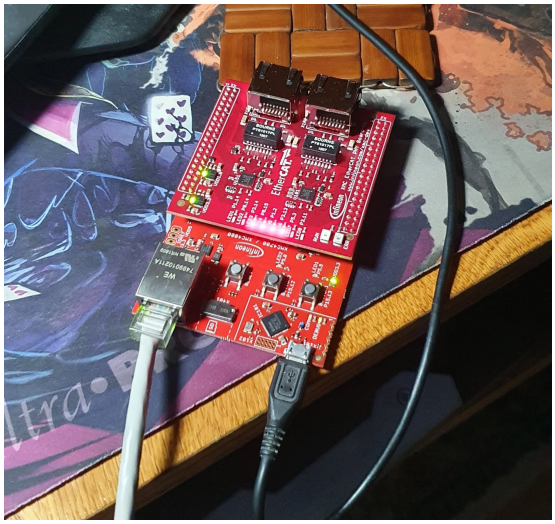
8. **Prezentare "Ce vreau sa demonstrez"**

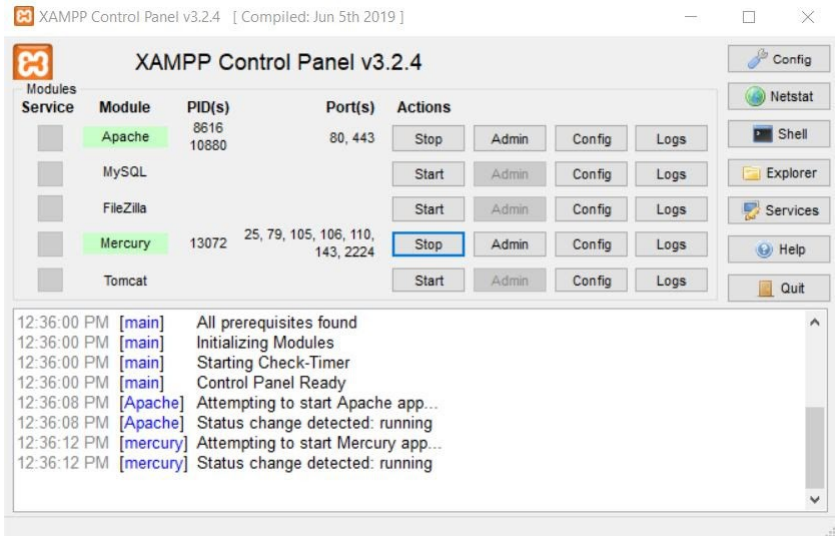
Vrem sa aratam functionalitatea incrementarii pe 8 biti de la valoarea 0 la valoarea maxima (255) cat si posibilitatea conexiunii unui dispozitiv la un server pentru a-i monitoriza functionalitatile.

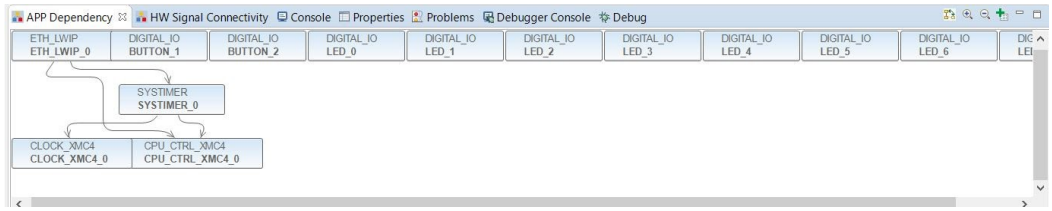
9. Secventa demonstrativa(schema, cod, video)

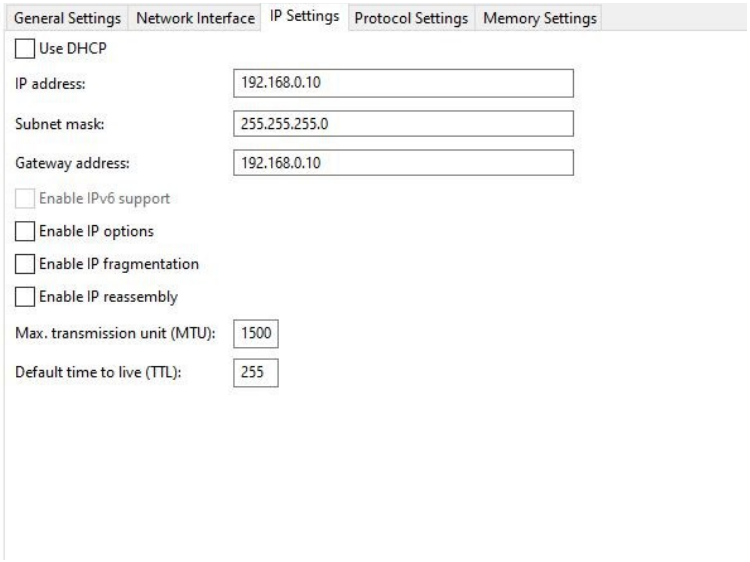
Video:

https://youtu.be/JDR6fkN3psM









Cod Dave:

```
#include <DAVE.h>

#define SERVER_IP_ADDR0    192U
#define SERVER_IP_ADDR1    168U
#define SERVER_IP_ADDR2    0U
#define SERVER_IP_ADDR3    5U

#define SERVER_HTTP_PORT   80U
#define DOMAIN_NAME "192.168.0.5"
#define DHCP_ENABLED 0U
#define DEVICE_ID 0U

void client_err(void *arg, err_t err);
void client_init(void);
err_t client_connected(void *arg, struct tcp_pcb *pcb, err_t err);
void client_close(struct tcp_pcb *pcb);
err_t client_sent(void *arg, struct tcp_pcb *pcb, u16_t len);
/*vector leduri*/
const DIGITAL_IO_t*
LED_X[8]={&LED_0,&LED_1,&LED_2,&LED_3,&LED_4,&LED_5,&LED_6,&LED_7};
extern struct netif xnetif;
/* Blocul de instructiuni de control pentru protocolul TCP */
struct tcp_pcb *pcb_send, *pcb_open;
/* Starea conexiunii */
uint8_t connection_ready=0,pcb_valid=0;
uint8_t counter=0;
void client_err(void *arg, err_t err)
{
```

```
  if (err == ERR_RST)
    pcb_valid=0;
  return;
}
/*initializam conexiunea la server*/
void client_init(void)
{
  struct ip_addr dest;
  IP4_ADDR(&dest, SERVER_IP_ADDR0, SERVER_IP_ADDR1, SERVER_IP_ADDR2,
SERVER_IP_ADDR3);
  if (pcb_open!=0)
    tcp_abort(pcb_open);
  pcb_open = tcp_new();
  tcp_err(pcb_open, client_err);
  tcp_bind(pcb_open, IP_ADDR_ANY, SERVER_HTTP_PORT);
  tcp_arg(pcb_open, NULL);
  tcp_connect(pcb_open, &dest, SERVER_HTTP_PORT, client_connected);
}

err_t client_connected(void *arg, struct tcp_pcb *pcb, err_t err)
{
  if (err==ERR_OK)
  {
    connection_ready=1;
    pcb_valid=1;
    pcb_send=pcb;
  }
  else
  {
    client_close(pcb);
    pcb_valid=0;
  }
  return err;
}
void client_close(struct tcp_pcb *pcb)
{
  tcp_arg(pcb, NULL);
  tcp_sent(pcb, NULL);
  tcp_abort(pcb);
```

```
}
err_t client_sent(void *arg, struct tcp_pcb *pcb, u16_t len)
{
  client_close(pcb);
  pcb_valid=0;
  client_init();
  return ERR_OK;
}
/*
 * Trimite header-ul HTTP si seteaza Led-urile;
 *
 * button1        - Seteaza pe 1 valoarea dupa ce a fost apasat butonul 1, sau 0 in cazul opus;
 * button2        - Seteaza pe 1 valoarea dupa ce a fost apasat butonul 2, sau 0 in cazul opus;
 * counter_state  - Starea clientului
*/
void send_data(uint8_t button1, uint8_t button2, uint8_t counter_state)
{
  static uint32_t connect_WD=0;
  char http_header[1024];
  sprintf(http_header,"GET /pushData.php?"
                      "device=%i&value=%i&button1=%i&button2=%i HTTP/1.1\r\n"
                      "Host: %s\r\n"
                      "Connection: keep-alive\r\n"
                      "Pragma: no-cache\r\n"
                      "Cache-Control: no-cache\r\n"
                      "Accept: text/html,application/xhtml+xml,"
                      "application/xml;q=0.9,image/webp,*/*;q=0.8\r\n"
                      "Accept-Encoding: gzip, deflate, sdch\r\n\r\n"
                      ,DEVICE_ID, counter_state, button1, button2, DOMAIN_NAME);
  if ((connection_ready==1)&&(pcb_send!=0))
  {
    connection_ready=0;
    connect_WD=0;
    tcp_sent(pcb_send, client_sent);
    tcp_write(pcb_send, (char*)&http_header, sizeof(http_header), 0);
    tcp_output(pcb_send);
    for (uint8_t bit=0; bit<8; bit++)
    {
      if (counter_state&(1<<bit))
```

```
      DIGITAL_IO_SetOutputLow(LED_X[bit]);
    else
      DIGITAL_IO_SetOutputHigh(LED_X[bit]);
   }
  }
  else
  {
   connect_WD++;
   if (connect_WD==10)
   {
    client_close(pcb_send);
    pcb_valid=0;
   }
  }
}
void tim_sys_check_timeouts_wrap(void *args)
{
  sys_check_timeouts();
}

/* Se incrementeaza counter-ul */
void tim_counter_increment(void *args)
{
  counter++;
}
void check_buttons_released(uint8_t *ptr_button1, uint8_t *ptr_button2, uint8_t reset)
{
  static uint8_t button1_state_up=0,button2_state_up=0,button1=0,button2=0;
  if (DIGITAL_IO_GetInput(&BUTTON_1)==0)
  {
   button1_state_up=1;
  }
  else
  {
   if (button1_state_up==1)
   {
    button1_state_up=0;
    button1=1;
   }
```

```
  }
  if (DIGITAL_IO_GetInput(&BUTTON_2)==0)
  {
    button2_state_up=1;
  }
  else
  {
    if (button2_state_up==1)
    {
      button2_state_up=0;
      button2=1;
    }
  }
  *ptr_button1=button1;
  *ptr_button2=button2;
  if (reset==1)
  {
  button1=0;
  button2=0;
  }

}

int main(void)
{
  DAVE_STATUS_t status;
  uint32_t timer_systimer_lwip, timer_systimer_counter;
  uint8_t button1=0,button2=0;
  uint8_t counter_old=0;
  status = DAVE_Init();
  if(status != DAVE_STATUS_SUCCESS)
  {
    XMC_DEBUG("DAVE APPs initialization failed\n");
    while(1U)
    {
    }
  }
  timer_systimer_lwip = SYSTIMER_CreateTimer(100000,
```

```
                                    SYSTIMER_MODE_PERIODIC , tim_sys_check_timeouts_wrap,0);
  SYSTIMER_StartTimer(timer_systimer_lwip);

  /* Timer pentru starea clientului */
  timer_systimer_counter = SYSTIMER_CreateTimer (500000,
                                    SYSTIMER_MODE_PERIODIC ,
                                    tim_counter_increment,0);
  SYSTIMER_StartTimer(timer_systimer_counter);
  while(1)
  {
    if (counter!=counter_old)
    {
      if ((connection_ready==0)&&(pcb_valid==0))
      {
#if DHCP_ENABLED==1
       /* Verificam daca adresa IP este inca valida prin DHCP */
       if ((xnetif.dhcp->state) == DHCP_BOUND)
#endif
         client_init();
      }
      else
      {
        check_buttons_released(&button1, &button2, 1);
        send_data(button1, button2, counter);
      }
      counter_old=counter;
    }
    check_buttons_released(&button1, &button2, 0);
  }
}
```

10. Concluzii

Cu ocazia realizarii acestui proiect am dobandit cunostinte noi in ceea ce priveste modul de utilizare al dispozitivelor hardware cat si modul de utilizare al programului DAVE. De asemenea, ne-am putut familiariza si cu software-ul XAMPP pentru realizarea unei conexiuni in reteaua locala.

11. Domenii de aplicabilitate

Acest proiect poate fi extins pentru monitorizarea dispozitivelor dintr-un Smart House prin

intermediul unui server.

12. Bibliografie

http://embedac.ro/SI/Lab/L3/Laborator3.htm

https://www.infineon.com/dgdl/Infineon-Board_User_Manual_XMC4700_XMC4800_Relax_Kit_Series-UM-v01_02-EN.pdf?fileId=5546d46250cc1fdf01513f8e052d07fc

https://www.youtube.com/watch?v=7prWcndctpQ

https://www.ionos.com/digitalguide/server/tools/xampp-tutorial-create-your-own-local-test-server/