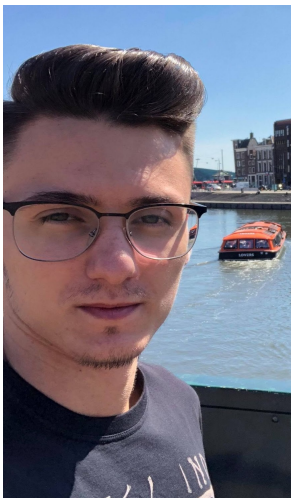
Titlul proiectului: LUFA implementation on your own USB Device

2. **Nume studenti:**

Aluculesei Pavel (1405A)

Gavriliuc Eduard Dan (1405B)

Popa Sebastian Paul (1405A)



3. **Email:**

pavel.aluculesei@student.tuiasi.ro
eduard-dan.gavriliuc@student.tuiasi.ro
sebastian-paul.popa@student.tuiasi.ro

4. **Rezumat:**

LUFA is a piece of software that helps you implement your own USB device. It

is not a programming interface (though of-course one use of LUFA could be to implement your own programmer or bootloader).

Proiect 1:

Titlu: Bootloader for USB Mass Storage
Rezumat: You will learn how to configure XMC4700 and implement the board as a media storage device. The bootloader executable lets you update the application image. For this you just need to drag & drop the binary image of the app to the USB mass storage device.

Proiect 2:

Titlu: USB HID (Human Interface Device)
Rezumat: You will learn how to configure XMC4700 and see the input button states of the board sent to the host and the LEDs on the board controlled from PC.

5. Resurse hardware utilizate

- XMC4700 Relax Kit
- 2 Micro USB Cable(Debugger & USB mass storage/HID connector)

6. Resurse software utilizate

- DAVE™
- Python
- Pwinusb

7. Prezentare “Ce vreau sa demonstrez”

Scopul primului proiect este de a face bootabila placuta si pentru a o folosi ca un media storage device, de exemplu o putem folosi pentru a face host propriului nostru server web (trebuie atasat un cablu Ethernet) sau pur si simplu pentru a stoca diferite informatii.

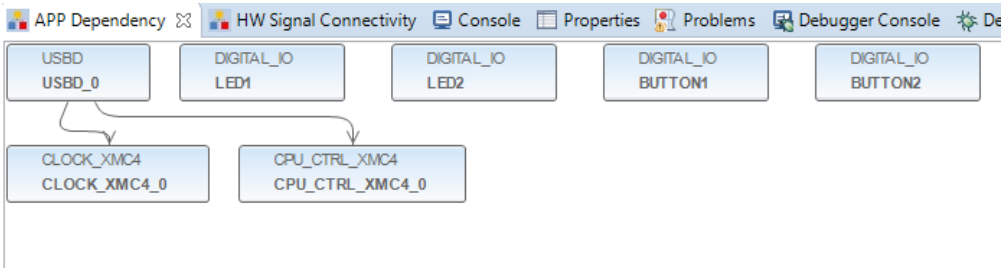
Scopul celui de al doilea proiect este de a arata interactiunea om - dispozitiv, care

fie se face prin diferite scripturi utilizand diferite programe, fie prin diferite componente hardware aferente placutei.

8. Secventa demonstrativa

Proiect 1:

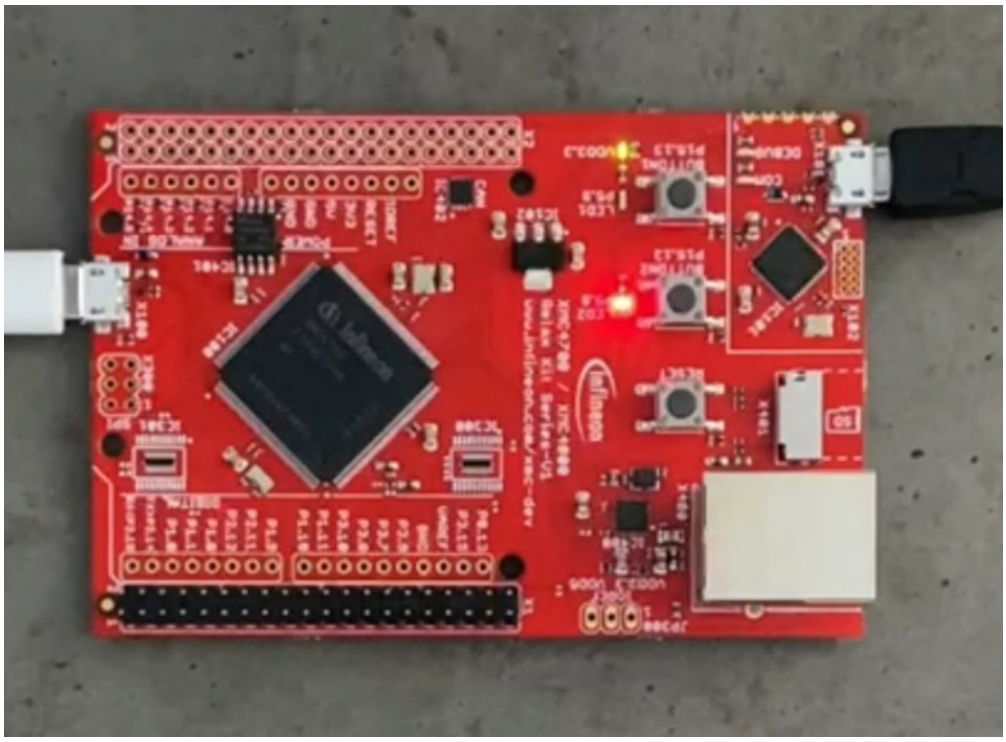
Video prezentare: https://streamable.com/0cmkeb



Cod:

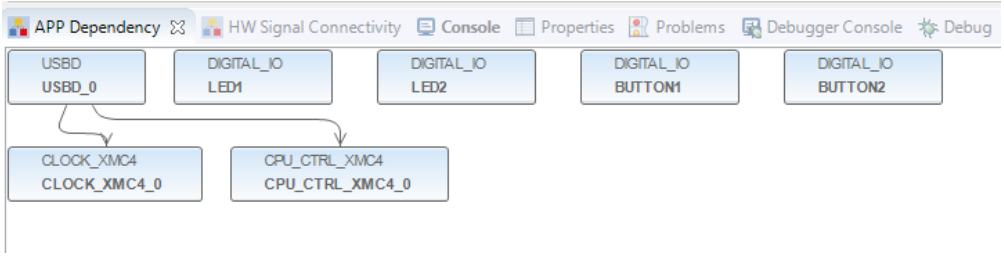
```c
#include <DAVE.h>
#include "USB_MS/Demos/Device/ClassDriver/MassStorage/Descriptors.h"
#include "VirtualFAT.h"
#include "MassStorage.h"

#define ABM_HEADER_MAGIC_KEY 0xA5C3E10F

typedef struct ABM_Header {
    uint32_t MagicKey; //0xA5C3E10F
    uint32_t StartAddress; //adresa de start
    uint32_t Length; //durata de rulare
    uint32_t ApplicationCRC32;
    uint32_t HeaderCRC32;
} ABM_Header_t;


static const ABM_Header_t  __attribute__((section(".flash_abm"), used))
ABM0_Header = {
    .MagicKey = ABM_HEADER_MAGIC_KEY,
    .StartAddress =   0x08010000,
    .Length = 0xFFFFFFFF,
    .ApplicationCRC32 = 0xFFFFFFFF,
    .HeaderCRC32 =  0xE176A4E6
};

void BL_FlashABM0_Restart(void)
{
  // reseteaza campul de RESET
  XMC_SCU_RESET_ClearDeviceResetReason();
  //setam ABM0 ca boot mod in  campul SWCON din registrul STCON
  XMC_SCU_SetBootMode(XMC_SCU_BOOTMODE_ABM0);
  // activare la apasarea butonului de RESET
  PPB->AIRCR = 1 << PPB_AIRCR_SYSRESETREQ_Pos |0x5FA<<PPB_AIRCR_VECTKEY_Pos | 0x1 << PPB_AIRCR_PRIGROUP_Pos;
}
```

```c
int main(void)
{
  //initializare program
  if(DAVE_Init() == DAVE_STATUS_FAILURE)
  {
    // errorhandler ul
    XMC_DEBUG(("DAVE APPs initialization failed\n"));
    while(1U)
    {
      //break
    }
  }

  if (DIGITAL_IO_GetInput(&BUTTON1))
  {
    // daca BUTTON1 nu este apasat, se restarteaza plcauta

    BL_FlashABM0_Restart();
  }
  else
  {
    // daca BUTTON1 este apasat, placuta intra in modul Media Storage

    // initializeaza modul virtual
    VFAT_Init();

    // intra in modul MS
    USB_Init();

    // proceseaza dimensiunea MS
    for( ;; )
    {
      MS_Device_USBTask(&Disk_MS_Interface);
    }
  }
  return 0;
}

void EVENT_USB_Device_Disconnect() {
  // daca se decupleaza, placuta de restarteaza

  BL_FlashABM0_Restart();
}
```



Proiect 2:

Video prezentare: https://streamable.com/82b547



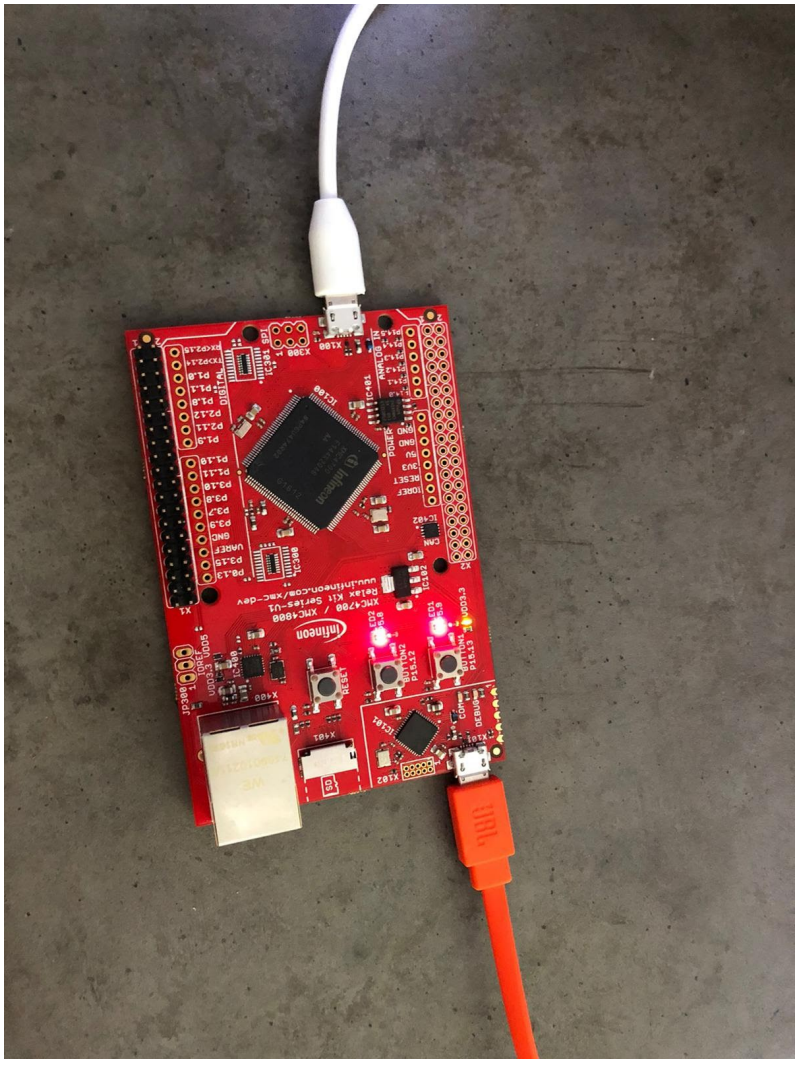
Cod:

```c
#include <DAVE.h>
#include "GenericHID.h"

#define REPORT_SIZE 2U

bool CALLBACK_HID_Device_CreateHIDReport(USB_ClassInfo_HID_Device_t* const HIDInterfaceInfo,
                                         uint8_t* const ReportID,
                                         const uint8_t ReportType,
                                         void* ReportData,
                                         uint16_t* const ReportSize)
{
  uint8_t* Data = (uint8_t*)ReportData;
  if (DIGITAL_IO_GetInput(&BUTTON1))
  {
    Data[0]=0;
  }
  else
  {
    Data[0]=1;
  }
  if (DIGITAL_IO_GetInput(&BUTTON2))
  {
    Data[1]=0;
  }
  else
  {
    Data[1]=1;
  }
  *ReportSize = REPORT_SIZE;

  return false;
}
```

```c
void CALLBACK_HID_Device_ProcessHIDReport(USB_ClassInfo_HID_Device_t* const HIDInterfaceInfo,
                                          const uint8_t ReportID,
                                          const uint8_t ReportType,
                                          const void* ReportData,
                                          const uint16_t ReportSize)
{
  uint8_t* Data       = (uint8_t*)ReportData;

  if (Data[0]==0)
  {
    DIGITAL_IO_SetOutputLow(&LED1);
  }
  else
  {
    DIGITAL_IO_SetOutputHigh(&LED1);
  }

  if (Data[1]==0)
  {
    DIGITAL_IO_SetOutputLow(&LED2);
  }
  else
  {
    DIGITAL_IO_SetOutputHigh(&LED2);
  }
}
```

```c
int32_t main(void)
{
  if(DAVE_Init() == DAVE_STATUS_FAILURE)
  {
    XMC_DEBUG(("DAVE APPs initialization failed\n"));
    while(1U)
    {

    }
  }

  USB_Init(REPORT_SIZE);

  for( ;; )
  {
    HID_Device_USBTask(&Generic_HID_Interface);
  }
}
```

```
C:\WINDOWS\py.exe
Sent LED Pattern      : [0, 0]
Sent LED Pattern      : [0, 1]
Sent LED Pattern      : [1, 0]
Sent LED Pattern      : [1, 1]
Received BUTTON State : [1, 0]
Received BUTTON State : [0, 0]
Sent LED Pattern      : [0, 0]
Received BUTTON State : [1, 0]
Sent LED Pattern      : [0, 1]
Received BUTTON State : [0, 0]
Sent LED Pattern      : [1, 0]
Received BUTTON State : [0, 1]
Received BUTTON State : [0, 0]
Sent LED Pattern      : [1, 1]
Sent LED Pattern      : [0, 0]
Sent LED Pattern      : [0, 1]
Sent LED Pattern      : [1, 0]
Sent LED Pattern      : [1, 1]
Sent LED Pattern      : [0, 0]
Sent LED Pattern      : [0, 1]
Sent LED Pattern      : [1, 0]
```



9. Concluzii:

Am reusit sa abordam 2 proiecte de pe site-ul celor de la Infineon, care nu au fost discutate in cadrul laboratoarelor de SI. Dupa parerea noastra, am invatat lucruri noi si interesante ce ulterior le putem folosi pentru alte proiecte ceva mai complexe, deoarece acest domeniu al IOT putem spune ca este intr-o continua evolutie.

10. **Domenii de aplicabilitate:**
    - Automatizari
    - Laboratoare

11.Bibliografie:

- https://www.infineon.com/dgdl/Infineon-Board_User_Manual_XMC4700_XMC4800_Relax_Kit_Series-UM-v01_02-EN.pdf?fileId=5546d46250cc1fdf01513f8e052d07fc
- https://www.infineon.com/cms/en/product/promopages/aim-mc/dave_downloads.html
- https://www.python.org/
- http://embedac.ro/SI/index.html

12.Contributii:

- Pavel: Elaborare documentatie
- Paul: Testare functionalitate, poze, video
- Eduard: Documentatie, research materiale, poze