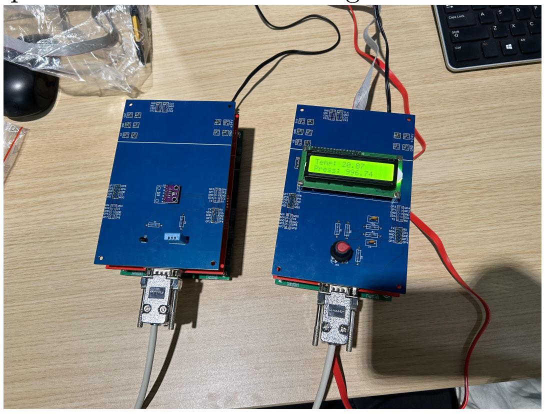
# PV284 Colloquium Project

Rebeka Černianska

January 2023

## 1 Project brief description

The aim of the project was to have a meteo station on a BluePill board measure metrics and share them with a BluePill station, which will display the measured data. As for the schematics, two nodes made up this solution. Node A was the meteo station, which collected data. Node B was the displaying station, which was responsible for displaying the data. Bellow, the final setup can be viewed in an image.



Here is the list of the used components for the project:

- 2 BluePill modules
- 2 Canbus communication modules
- application board with an LCD display
- application board with a meteo station - includes temperature meter DS18B20 and a barometric pressure measurement BME-280

## 2 Project solution

The goal of the project was to further develop the setup of one of the labs, which was focused on the meteo station provided by one of the application boards. The data which is displayed on the LCD display is the current temperature and atmospheric pressure of the meteo station's location. The data can further be changed by the rotary encoder button, which upon pressing, will change the unit system of the temperature. Therefore, the temperature can be viewed in both Celsius and Fahrenheit. The temperature and pressure are displayed at the same time and update regularly.

Node A was responsible for the collection of data, which was done by the temperature meter DS18B20 and a barometric pressure measurement BME-280 sensors, which are a part of the application board with a meteo station. These sensors were utilized to measure the data, which would be later sent to the displaying node. For the transfer of data, the Canbus was used, which sent the data from one node, to the other. On the displaying node, the LCD display was used, along with the rotary encoder. Important libraries which were utilized were the DallasTemperature and Adafruit_BMP280 libraries. Their functions were responsible for collecting the data, as well as transfering the data among different data units. There were only minor problems along the way, which are not notable.

## 3 Conclusion

In conclusion, the project can be used as a basis for a larger project, which would be sending data to cloud. This project aimed to collect and display more data locally, which can be used as a simple home solution.

## 4 Appendixes

### 4.1 Software of node A

```
/******************************************************************/
// First we include the libraries
#include <OneWireSTM.h>
#include <DallasTemperature.h>
/******************************************************************/
// Data wire is plugged into pin PA15 on the BluePill
#define ONE_WIRE_BUS PA15

// https://github.com/coryjfowler/MCP_CAN_lib
#include <mcp_can.h>
#include <SPI.h>

MCP_CAN CAN0(PA4);   // Set CS to pin PA4
/******************************************************************/
// Setup a oneWire instance to communicate with any OneWire devices
// (not just Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);
/******************************************************************/
// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);
/******************************************************************/
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
Adafruit_BMP280 bme;
#define SEALEVELPRESSURE_HPA (1013.25)
bool status;

float tempF;
float tempBME;
float pressBME;
float altBME;

void setup(void)
{
  afio_cfg_debug_ports(AFIO_DEBUG_SW_ONLY);
  Serial.begin(9600);

  // Initialize MCP2515 running at 16MHz with a baudrate of 500kb/s and the masks and
      filters disabled.
  if(CAN0.begin(MCP_ANY, CAN_500KBPS, MCP_16MHZ) == CAN_OK) Serial.println("MCP2515
      Initialized Successfully!");
  else Serial.println("Error Initializing MCP2515...");

  CAN0.setMode(MCP_NORMAL); // Change to normal mode to allow messages to be
      transmitted

  status = bme.begin(0x76); //adafruit
}

byte data[8] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07};
float temp;

void loop(void)
{
  // call sensors.requestTemperatures() to issue a global temperature
  // request to all devices on the bus
  /********************************************************************/
  Serial.print(" Requesting temperatures...");
  sensors.requestTemperatures(); // Send the command to get temperature readings
  Serial.println("DONE");
  /********************************************************************/
  Serial.print("Temperature is: ");
  temp = sensors.getTempCByIndex(0);
  Serial.print(temp);

  delay(1000);
  byte bytesTemp[4];
  byte bytesPress[4];
  byte bytesAlt[4];

  //original Dallas Temperature temp collection
  *((float *)bytesTemp) = temp;
  // send data: ID = 0x100, Standard CAN Frame, Data length = 8 bytes, 'data' = array
      of data bytes to send
  byte sndStat = CAN0.sendMsgBuf(0x100, 0, 4, bytesTemp);
  if(sndStat == CAN_OK){
    Serial.println("Message Sent Successfully!");
  } else {
    Serial.println("Error Sending Message...");
  }
  delay(1000);

//  tempBME = bme.readTemperature();
  //AdaFruit Pressure reading/collection
  *((float *)bytesPress) = bme.readPressure() / 100.0F;


  // send data: ID = 0x100, Standard CAN Frame, Data length = 8 bytes, 'data' = array
      of data bytes to send
  sndStat = CAN0.sendMsgBuf(0x100, 0, 4, bytesPress);
  if(sndStat == CAN_OK){
    Serial.println("Message Sent Successfully!");
  } else {
    Serial.println("Error Sending Message...");
  }

  *((float *)bytesAlt) = bme.readAltitude(SEALEVELPRESSURE_HPA);}
```

## 4.2 Software of node B

```
#include <Wire.h>
#include <DallasTemperature.h>

// CAN Receive Example

// https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library
#include <LiquidCrystal_I2C.h>

// https://github.com/coryjfowler/MCP_CAN_lib
#include <mcp_can.h>
#include <SPI.h>

//pins for rotary encoder
#define CLK PB4
#define DT PB5
#define SW PA15


LiquidCrystal_I2C lcd(0x3F, 16, 2);
long unsigned int rxId;
unsigned char len = 0;
unsigned char rxBuf[8];
char msgString1[128];                   // Array to store serial string
char msgString2[128];

float tempF;
int counter = 0;
float old_temp;
float old_pressure;
int country = 1;

int lastStateCLK;
int currentStateCLK;
unsigned long LastButtonPress;

#define CAN0_INT PB15                         // Set INT to pin PB5
MCP_CAN CAN0(PA4);                          // Set CS to pin PA4


void setup()
{ afio_cfg_debug_ports(AFIO_DEBUG_SW_ONLY);
  Serial.begin(9600);

  // Initialize MCP2515 running at 16MHz with a baudrate of 500kb/s and the masks and
      filters disabled.
  if(CAN0.begin(MCP_ANY, CAN_500KBPS, MCP_16MHZ) == CAN_OK)
    Serial.println("MCP2515 Initialized Successfully!");
  else
    Serial.println("Error Initializing MCP2515...");

  CAN0.setMode(MCP_NORMAL);                // Set operation mode to normal so the
      MCP2515 sends acks to received data.

  pinMode(CAN0_INT, INPUT);                       // Configuring pin for /INT input
  Serial.println("MCP2515 Library Receive Example...");

  // init LCD
  lcd.begin();

  pinMode(CLK,INPUT);
  pinMode(DT,INPUT);
  pinMode(SW, INPUT_PULLUP);
  lastStateCLK = digitalRead(CLK);
}

void loop()
{
  float temp;
  float pressure;

  currentStateCLK = digitalRead(CLK);
  // Remember last CLK state
  lastStateCLK = currentStateCLK;

  int btnState = digitalRead(SW);
  if (btnState == LOW) {
  if (millis() - LastButtonPress > 50)
  {
    country = country * (-1);
    Serial.print("BUTTON PRESSED");
    Serial.print(country);
  }
  LastButtonPress = millis();
  }

  if(!digitalRead(CAN0_INT))                        // If CAN0_INT pin is low, read
      receive buffer
  {
    CAN0.readMsgBuf(&rxId, &len, rxBuf);  // Read data: len = data length, buf = data
        byte(s)

//    Serial.print("First reading: ");
//    Serial.print(*(float*)rxBuf);
//    Serial.println();

    if (!counter)
    {
      temp = *(float*)rxBuf;
      old_temp = temp;
      pressure = old_pressure;
      counter++;
    }
    else
    {
      pressure = *(float*)rxBuf;
      old_pressure = pressure;
      temp = old_temp;
      counter--;
    }

    if (country == -1)
      temp = DallasTemperature::toFahrenheit(temp);
//    Serial.print("Temp in Fahrenheit: ");
//    Serial.print(tempF);

    // Arduino's sprintf does not support floats
    sprintf(msgString1, "Temp: %d.%d", (int)temp, (int)(temp * 100) % 100);
    sprintf(msgString2, "Press: %d.%d", (int)pressure, (int)(pressure * 100) % 100);

    Serial.println();

    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print(msgString1);

    //lcd.clear();
    lcd.setCursor(0,1);
    lcd.print(msgString2);
  }
}

/*********************************************************
  END FILE
**********************************************************/
```