|  |  |
| --- | --- |
| 년도-학기 | 2021년 1학기 |
| 과목명 | 임베디드시스템설계 |

|  |  |
| --- | --- |
| **LAB번호** | **제목** |
| 13 | CAN |

|  |  |
| --- | --- |
| 실험 일자 | 2021년 6월 11일 |
| 제출자 이름 | 강\*\* |
| 제출자 학번 | 201803\*\*\*\* |
| 팀원 이름 |  |
|  |  |

**Chapter 1. 프로그램의 동작 방식 설명**

프로그램의 큰 구조는 다음과 같다.

STM32F407 Board

+ DHT22 센서

(온도, 습도 측정)

CAN bus

Raspberry Pi

(온, 습도 데이터 수신,

문자열 변환)

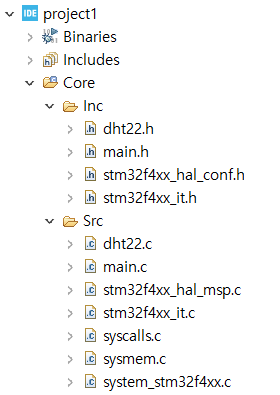
Ubidots.com

Socket 통신, Internet

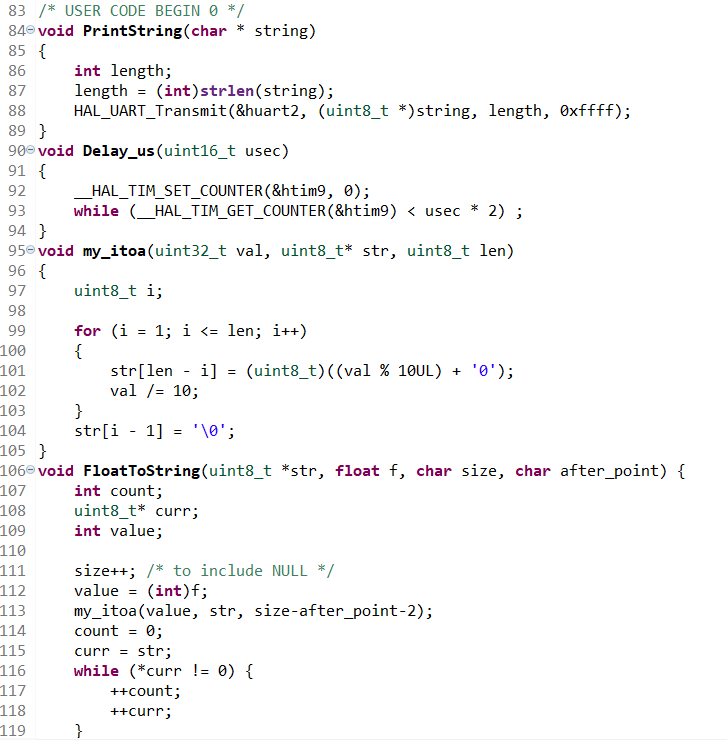
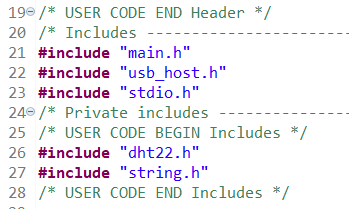
STM32F407 보드는 DHT22 센서를 연결하여 온도와 습도를 입력 받는다. 그리고 측정된 온도와 습도 정보는 CAN bus를 통해 Raspberry Pi로 전달된다. 측정된 온도와 습도는 3자리 정수로 입력 받기 때문에 백과 십의자리 숫자를 하나의 바이트에, 일의 자리 숫자를 다른 하나의 바이트에 넘겨 보내주었다. 각 자리의 숫자를 전달받은 Raspberry Pi에서는 이를 문자열로 저장하여 ubidots와의 소켓 통신으로 온도와 습도를 전달한다. Ubidots에서 받은 온, 습도 정보를 확인한다.

STM32F407 보드와 Raspberry pi의 코드를 바탕으로 살펴보면 다음과 같다.

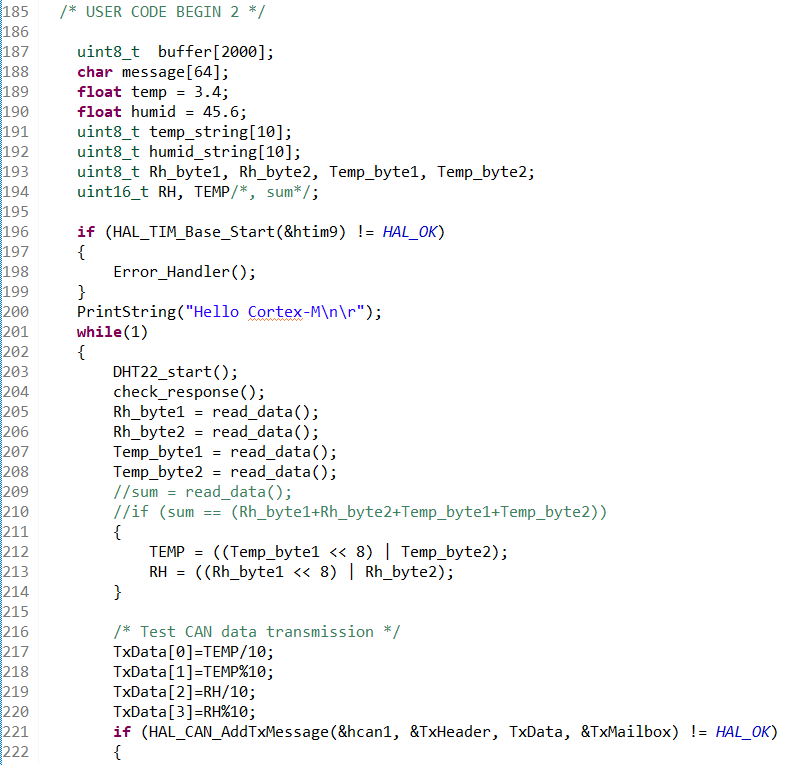
STM32F407 보드에서 온도와 습도를 측정하기 위해 dht22.h를 프로젝트의 Inc에 그리고 dht22.c 파일도 Src에 복사해 붙여 넣는다. 이를 통해 온도와 습도를 읽을 수 있다.

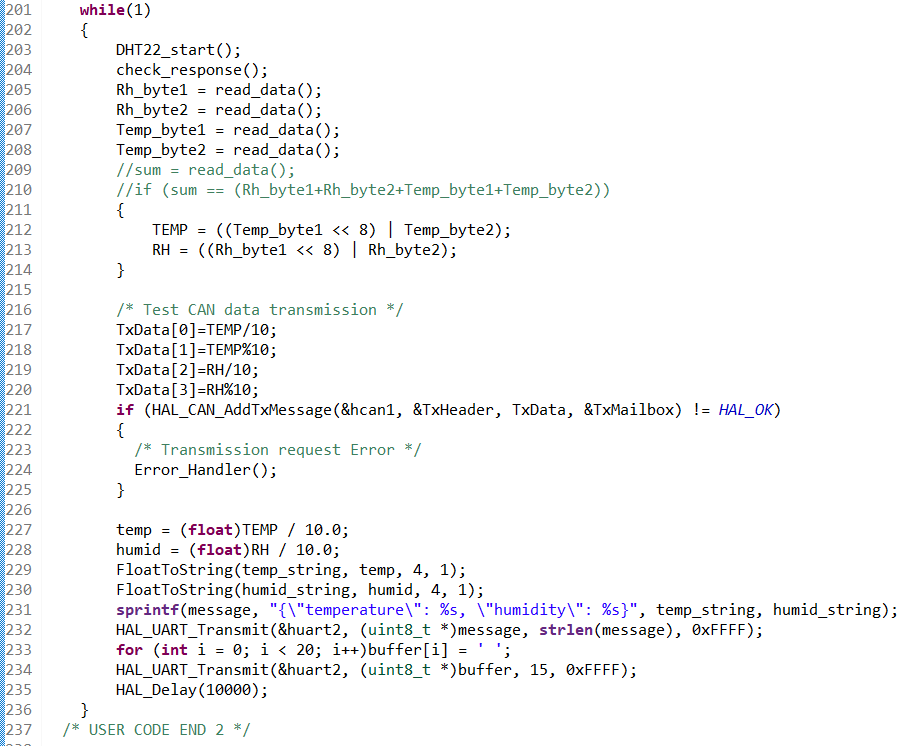




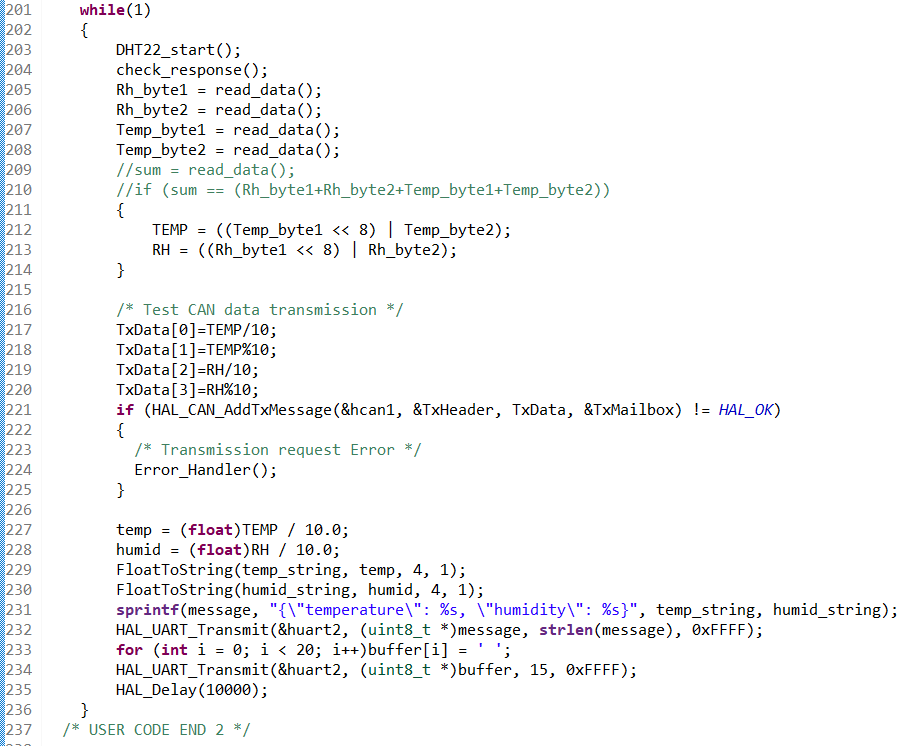
그 다음 필요한 헤더파일과 이후 구현에 필요한 함수를 추가한다.

main 함수에 온도와 습도를 읽고, can 통신을 하기 위한 코드를 추가하였다. 필요한 변수를 추가하였고, 무한 반복문에서 read\_data를 통해 온도와 습도의 정보를 바이트로 읽고 비트 연산을 통해 정수형의 TEMP(온도), RH(습도)를 얻어낸다.

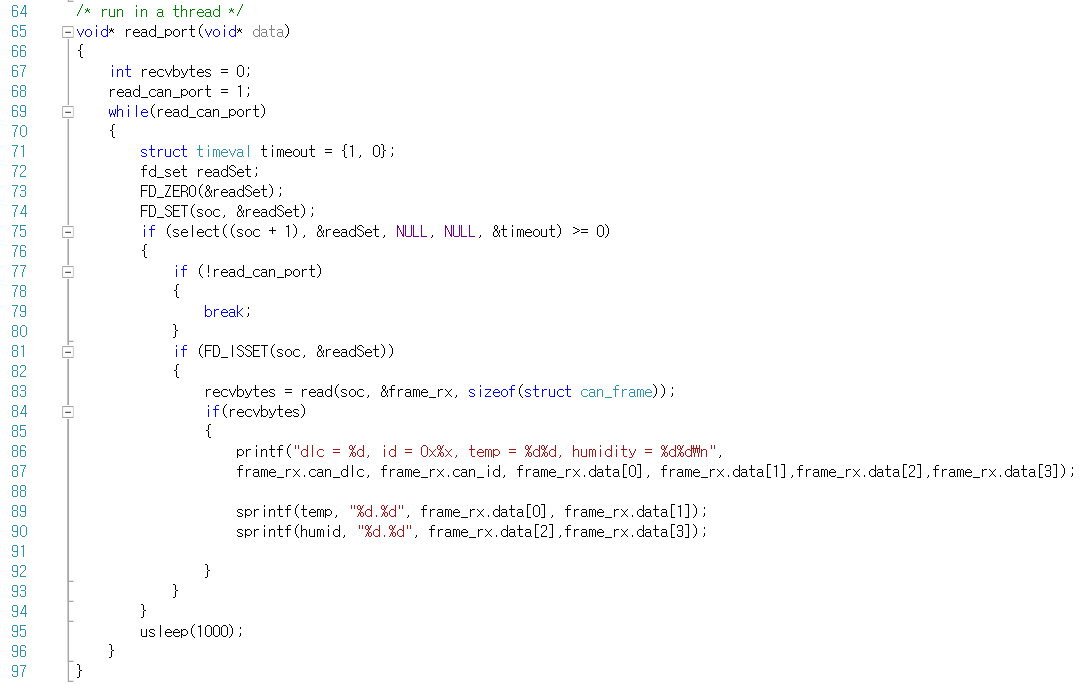




이후 TEMP와 RH를 백의 자리와 십의 자리, 그리고 일의 자리로 나눠서 TxData[]에 저장하고 CAN bus를 통해 Raspberry Pi로 데이터를 전달하였다. 센서로 읽은 데이터는 float형으로 변환하고 HAL\_UART\_Transmit 함수를 통해 확인해보았다.



Raspberry Pi에서는 SocketCANexample.c 코드를 참고해 port와 관련된 함수를 생성하였고, CAN을 통해 데이터를 수신하는 read\_port 함수는 별도의 thread로 생성하였다. 수신된 온도와 습도 값은 temp와 humid라는 글로벌 변수에 문자열로 저장하였다.



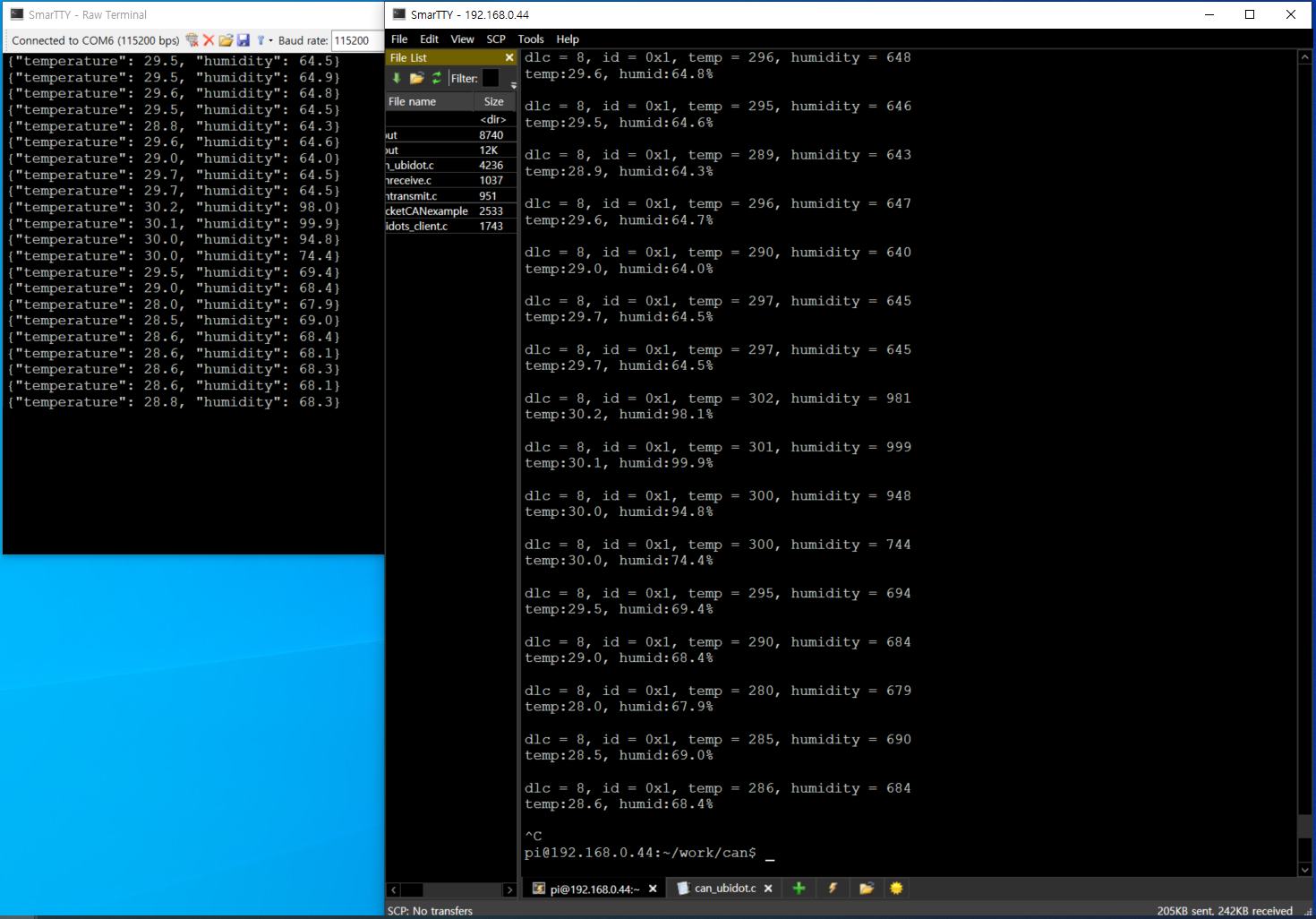
main 함수에서는 can 통신을 위해 port를 열고 데이터 수신을 하는 thread를 생성하였다. 그리고 수신한 문자열로 변환한 온도와 습도 값을 ubidots로 전송하여 과제를 수행한다.



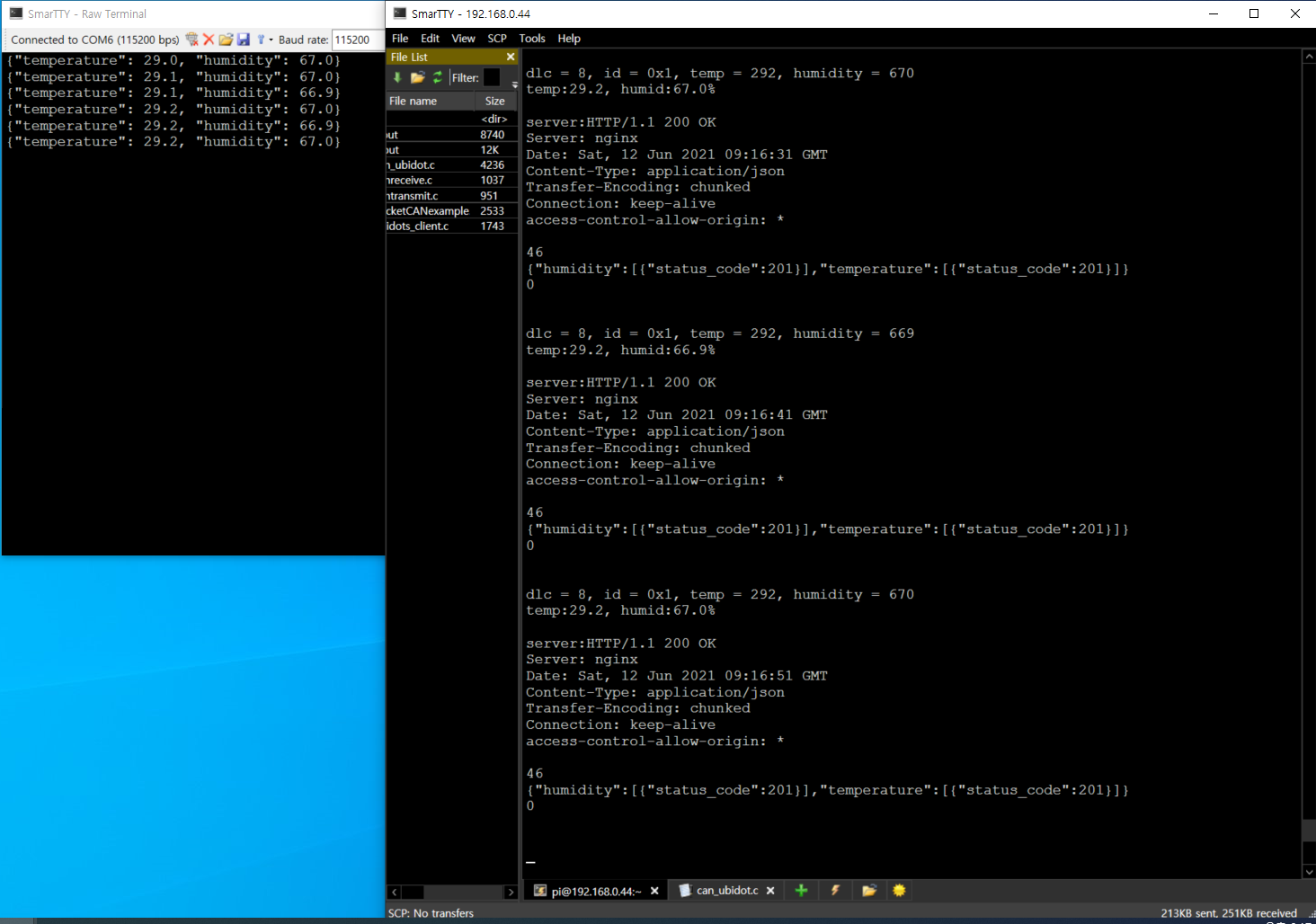


**Chapter 2. 결과**

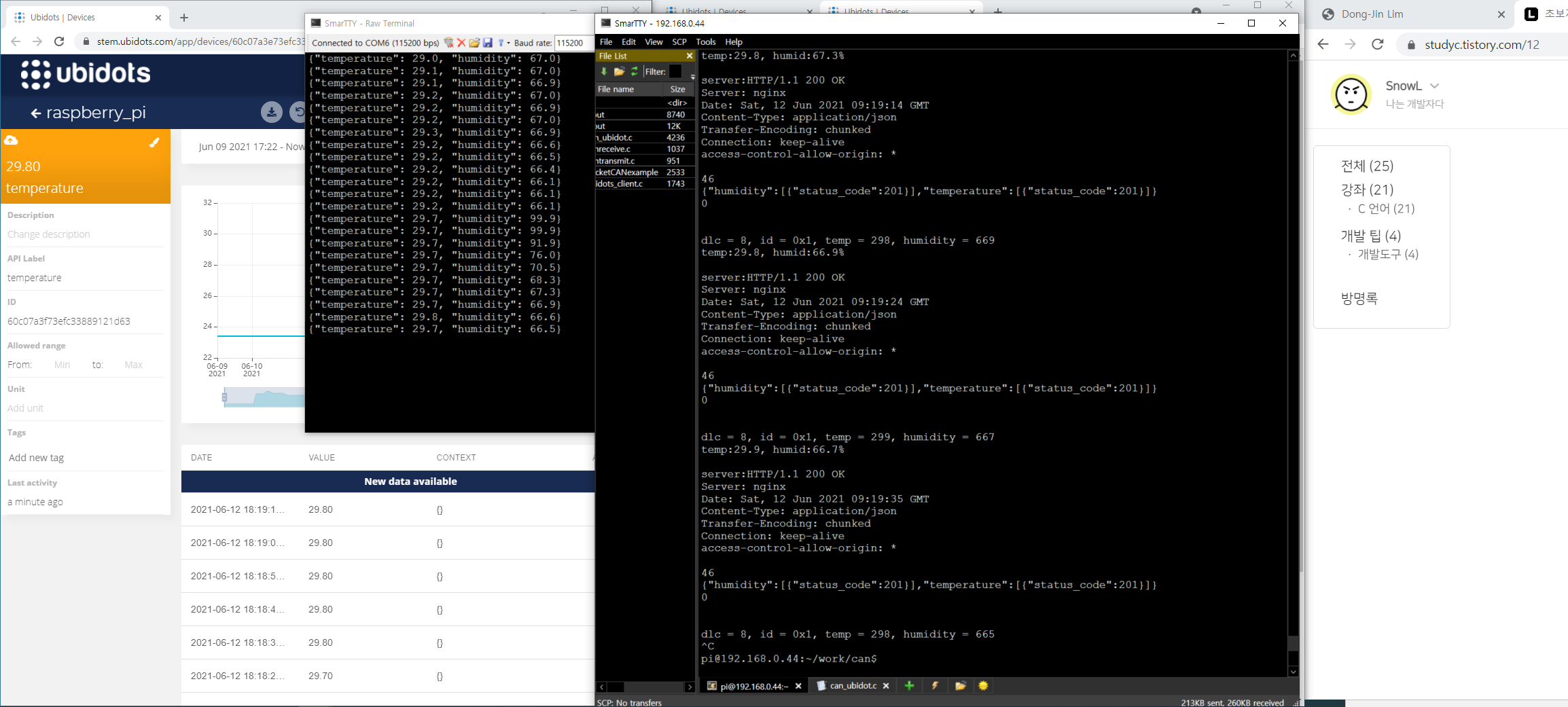
왼쪽은 센서를 통해 읽은 데이터 값을 출력한 창이고, 오른쪽은 raspberry pi에서 수신한 데이터 값을 출력한 것이다. 최대한 많이 비교하기 위해 데이터 값만 출력하도록 하였다. 거의 대부분이 같은 결과를 출력하지만 간혹 0.1 차이가 나는 경우도 발생하였다. 각 코드에서 STM32F407 보드는 HAL\_Delay(10000)마다 값을 출력하고, Raspberry Pi가 read\_port thread를 수행할 때 usleep(1000)이 존재하기 때문에 간혹 차이가 발생한 것이라고 생각한다.



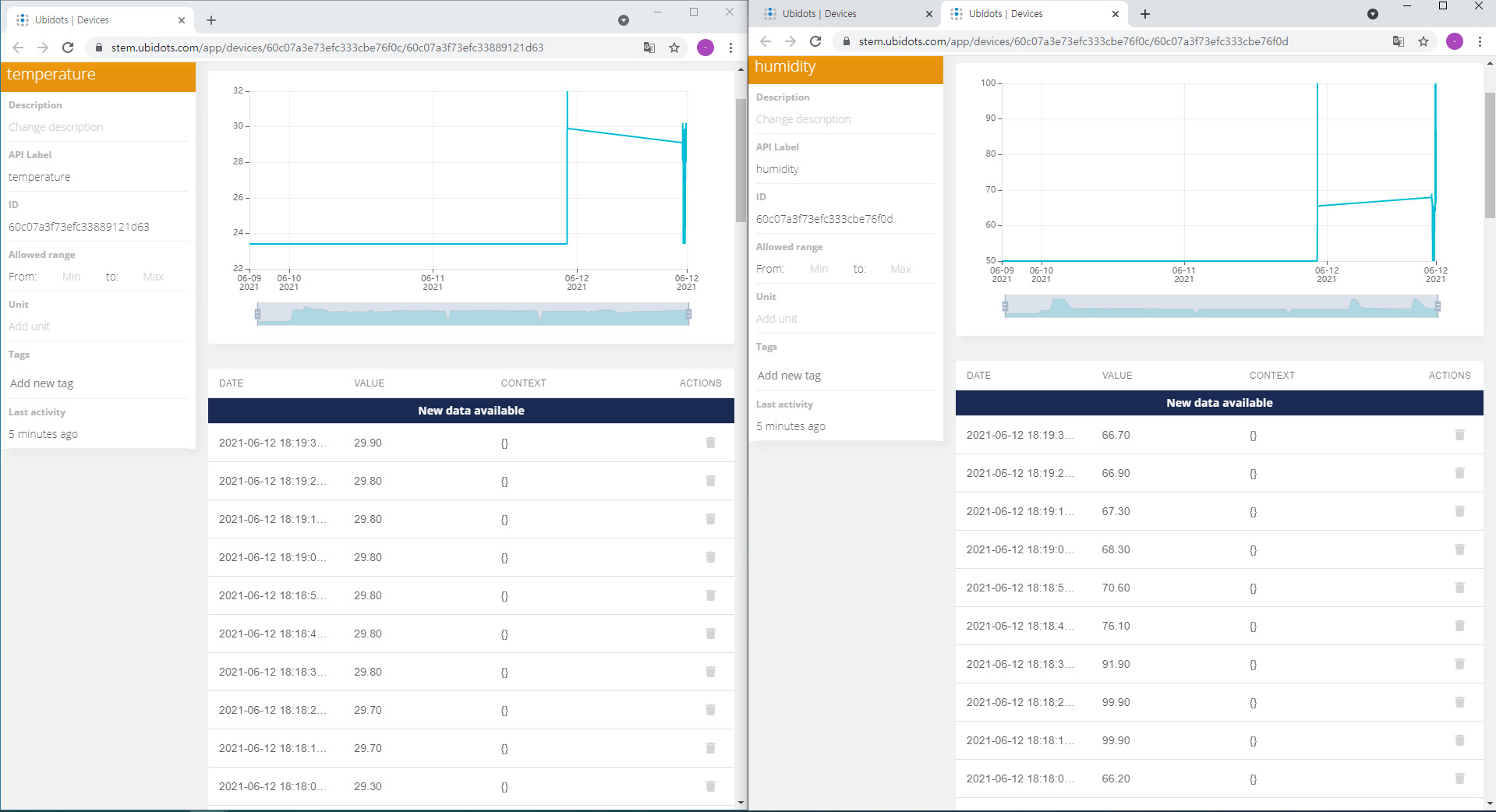
추가적으로 터미널에서 ubidots와의 통신이 잘 되고 있음은 아래의 사진에서 확인할 수 있다.

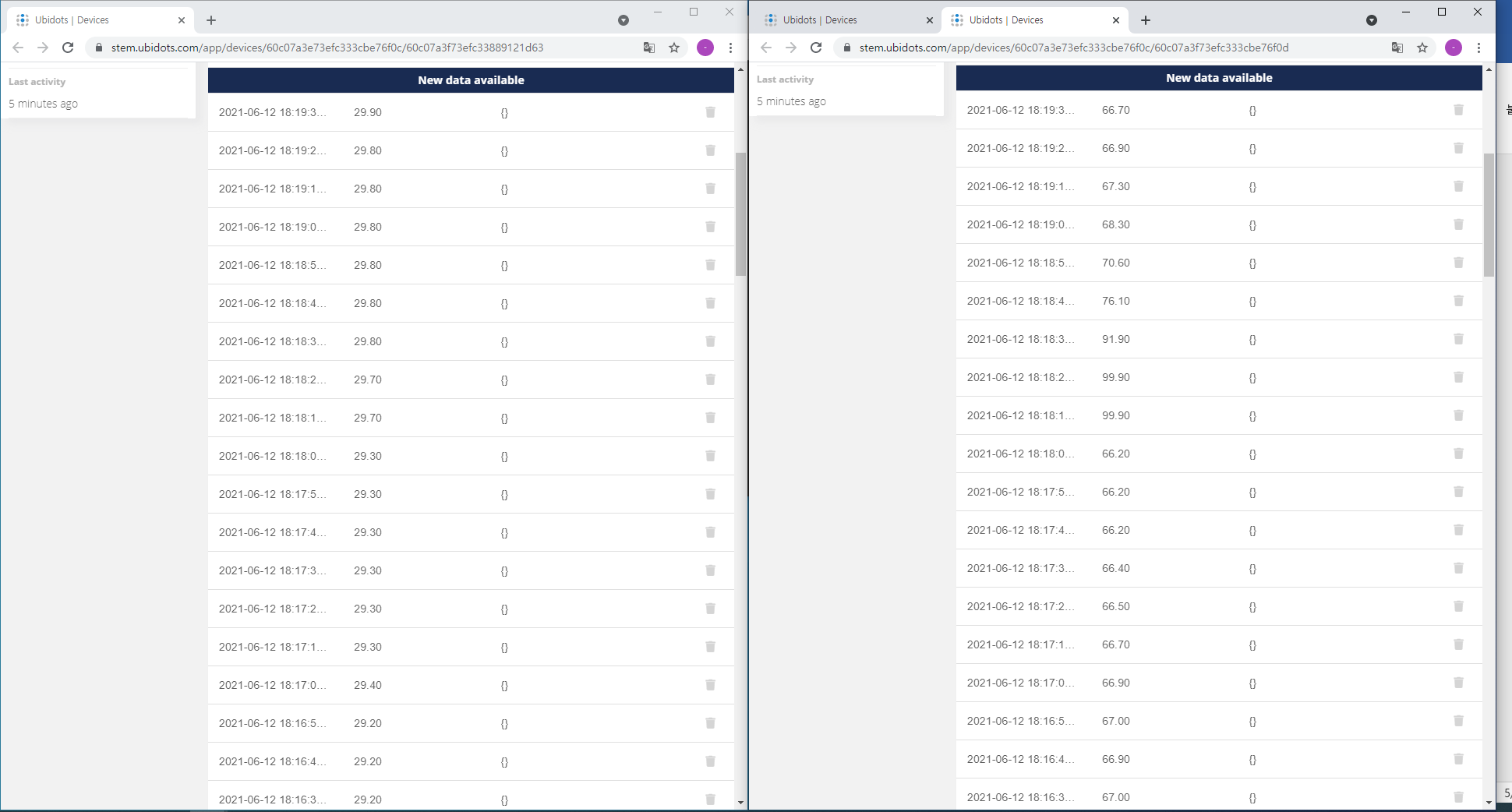


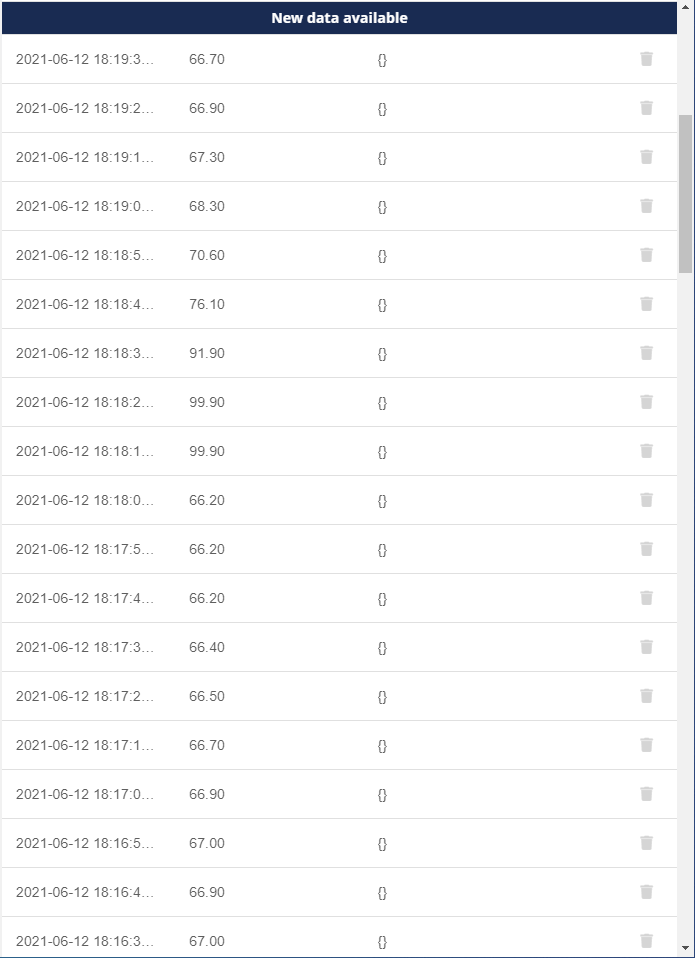
이 상태에서 ubidots에 전달된 데이터를 확인해보았다. 우선 센서로 입력 받은 온도와 습도 값과 라즈베리 파이가 전달 받은 데이터 상황이다.



앞서 말해듯이 출력되는 시간의 차이로 인해 0.1정도의 차이가 발생할 수 있음을 참고하고 결과를 보면, 잘 진행되었음을 확인할 수 있다. 또한, 데이터 변화를 확인하게 위해 입김을 불고 선풍기를 틀어 습도의 변화가 크게 나타나도록 하였다. 그래프와 데이터 값들을 보면 습도가 67% 근처의 습도에서 99.9%까지 올라갔다가 다시 내려가 66.7%을 마지막으로 잘 전송된 것을 확인할 수 있다. 확인을 돕기 위해 습도 데이터만 확대한 사진을 다음 페이지에 첨부하였다.



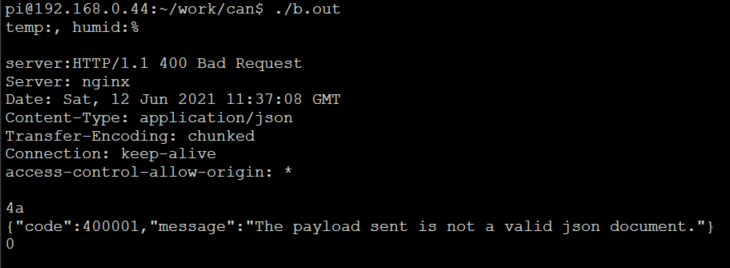




**Chapter 3. 결론 및 Discussion**

우선, 과제가 요구하는 것들을 잘 수행하였다. STM32F407 보드에서 DHT22 센서로 온도와 습도도 제대로 측정하였고, 보드와 raspberry pi 사이의 CAN통신과 raspberry pi와 ubidots의 socket 통신도 원활하게 진행되었다.

다만 raspberry pi에서 파일을 실행하면 첫번째로 ubidots에 데이터를 보내는 부분에서 아래 사진처럼 오류가 발생한다. 아직 데이터를 전달받지 못한 상태에서 ubidots에 데이터를 전송하려해서 생긴 문제라고 생각하였고, 이를 해결하고자 온, 습도의 문자열 변수를 특정 값으로 초기화 해보았으나 같은 결과가 나왔다. 오류 내용을 살펴보았을 때 전달된 데이터를 문자열 message로 저장하는 과정에 문제가 있는 것으로 보았는데 원인이 무엇인지 정확히 알아내지 못했고, 이를 해결하지 못해서 조금 아쉬움이 남는다. 그래도 처음 코드를 실행했을 때 이후로는 문제없이 데이터를 수신하고 ubidots 사이트로 전달하기 때문에 괜찮다고 생각했다.



센서도 사용하고 보드도 두 개 사용해서 처음에는 막막하기도 하고 어려울 것이라고 생각했지만, 예제로 해보고 배운 것을 차근차근 적용해보니 크게 어려운 부분은 없었다. 마지막 프로젝트 과제인만큼 지금껏 배운 내용들을 모두 고려해볼 수 있었던 과제였고, 만족스러운 결과를 보였기 때문에 한 학기를 잘 마무리 한 것 같다.

**부록**

부록으로 코드를 첨부하지만 편의를 위해 stm32 프로젝트 폴더와 raspberry pi에서 사용한 소스코드를 별도로 첨부해 제출하겠습니다.

**<STM32F407 보드의 main.c>**

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.

\* All rights reserved.</center></h2>

\*

\* This software component is licensed by ST under Ultimate Liberty license

\* SLA0044, the "License"; You may not use this file except in compliance with

\* the License. You may obtain a copy of the License at:

\* www.st.com/SLA0044

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#include "usb\_host.h"

#include "stdio.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

#include "dht22.h"

#include "string.h"

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

CAN\_HandleTypeDef hcan1;

I2C\_HandleTypeDef hi2c1;

I2S\_HandleTypeDef hi2s3;

SPI\_HandleTypeDef hspi1;

TIM\_HandleTypeDef htim9;

UART\_HandleTypeDef huart2;

UART\_HandleTypeDef huart3;

/\* USER CODE BEGIN PV \*/

CAN\_TxHeaderTypeDef TxHeader;

CAN\_RxHeaderTypeDef RxHeader;

uint8\_t TxData[8];

uint8\_t RxData[8];

uint32\_t TxMailbox;

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_I2C1\_Init(void);

static void MX\_I2S3\_Init(void);

static void MX\_SPI1\_Init(void);

static void MX\_CAN1\_Init(void);

static void MX\_USART2\_UART\_Init(void);

static void MX\_USART3\_UART\_Init(void);

static void MX\_TIM9\_Init(void);

void MX\_USB\_HOST\_Process(void);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

void PrintString(char \* string)

{

int length;

length = (int)strlen(string);

HAL\_UART\_Transmit(&huart2, (uint8\_t \*)string, length, 0xffff);

}

void Delay\_us(uint16\_t usec)

{

\_\_HAL\_TIM\_SET\_COUNTER(&htim9, 0);

while (\_\_HAL\_TIM\_GET\_COUNTER(&htim9) < usec \* 2) ;

}

void my\_itoa(uint32\_t val, uint8\_t\* str, uint8\_t len)

{

uint8\_t i;

for (i = 1; i <= len; i++)

{

str[len - i] = (uint8\_t)((val % 10UL) + '0');

val /= 10;

}

str[i - 1] = '\0';

}

void FloatToString(uint8\_t \*str, float f, char size, char after\_point) {

int count;

uint8\_t\* curr;

int value;

size++; /\* to include NULL \*/

value = (int)f;

my\_itoa(value, str, size-after\_point-2);

count = 0;

curr = str;

while (\*curr != 0) {

++count;

++curr;

}

if (count + 1 >= size) {

str[size - 1] = 0;

return;

}

str[count++] = '.';

++curr;

f = f - (float)value;

while (count + 1 < size) {

f \*= 10;

++count;

}

value = (int)f;

my\_itoa(value, curr, after\_point);

str[size - 1] = 0;

count = 0;

while (\*(str + count) == '0')

{

\*(str + count) = ' ';

count++;

}

return;

}

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_I2C1\_Init();

MX\_I2S3\_Init();

MX\_SPI1\_Init();

MX\_USB\_HOST\_Init();

MX\_CAN1\_Init();

MX\_USART2\_UART\_Init();

MX\_USART3\_UART\_Init();

MX\_TIM9\_Init();

/\* USER CODE BEGIN 2 \*/

uint8\_t buffer[2000];

char message[64];

float temp = 3.4;

float humid = 45.6;

uint8\_t temp\_string[10];

uint8\_t humid\_string[10];

uint8\_t Rh\_byte1, Rh\_byte2, Temp\_byte1, Temp\_byte2;

uint16\_t RH, TEMP/\*, sum\*/;

if (HAL\_TIM\_Base\_Start(&htim9) != HAL\_OK)

{

Error\_Handler();

}

PrintString("Hello Cortex-M\n\r");

while(1)

{

DHT22\_start();

check\_response();

Rh\_byte1 = read\_data();

Rh\_byte2 = read\_data();

Temp\_byte1 = read\_data();

Temp\_byte2 = read\_data();

//sum = read\_data();

//if (sum == (Rh\_byte1+Rh\_byte2+Temp\_byte1+Temp\_byte2))

{

TEMP = ((Temp\_byte1 << 8) | Temp\_byte2);

RH = ((Rh\_byte1 << 8) | Rh\_byte2);

}

/\* Test CAN data transmission \*/

TxData[0]=TEMP/10;

TxData[1]=TEMP%10;

TxData[2]=RH/10;

TxData[3]=RH%10;

if (HAL\_CAN\_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailbox) != HAL\_OK)

{

/\* Transmission request Error \*/

Error\_Handler();

}

temp = (float)TEMP / 10.0;

humid = (float)RH / 10.0;

FloatToString(temp\_string, temp, 4, 1);

FloatToString(humid\_string, humid, 4, 1);

sprintf(message, "{\"temperature\": %s, \"humidity\": %s}", temp\_string, humid\_string);

HAL\_UART\_Transmit(&huart2, (uint8\_t \*)message, strlen(message), 0xFFFF);

for (int i = 0; i < 20; i++)buffer[i] = ' ';

HAL\_UART\_Transmit(&huart2, (uint8\_t \*)buffer, 15, 0xFFFF);

HAL\_Delay(10000);

}

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

/\* USER CODE END WHILE \*/

MX\_USB\_HOST\_Process();

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

RCC\_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLM = 8;

RCC\_OscInitStruct.PLL.PLLN = 336;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 7;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV4;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_5) != HAL\_OK)

{

Error\_Handler();

}

PeriphClkInitStruct.PeriphClockSelection = RCC\_PERIPHCLK\_I2S;

PeriphClkInitStruct.PLLI2S.PLLI2SN = 192;

PeriphClkInitStruct.PLLI2S.PLLI2SR = 2;

if (HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInitStruct) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief CAN1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_CAN1\_Init(void)

{

/\* USER CODE BEGIN CAN1\_Init 0 \*/

CAN\_FilterTypeDef sFilterConfig;

/\* USER CODE END CAN1\_Init 0 \*/

/\* USER CODE BEGIN CAN1\_Init 1 \*/

/\* USER CODE END CAN1\_Init 1 \*/

hcan1.Instance = CAN1;

hcan1.Init.Prescaler = 6;

hcan1.Init.Mode = CAN\_MODE\_NORMAL;

hcan1.Init.SyncJumpWidth = CAN\_SJW\_1TQ;

hcan1.Init.TimeSeg1 = CAN\_BS1\_9TQ;

hcan1.Init.TimeSeg2 = CAN\_BS2\_4TQ;

hcan1.Init.TimeTriggeredMode = DISABLE;

hcan1.Init.AutoBusOff = DISABLE;

hcan1.Init.AutoWakeUp = DISABLE;

hcan1.Init.AutoRetransmission = DISABLE;

hcan1.Init.ReceiveFifoLocked = DISABLE;

hcan1.Init.TransmitFifoPriority = DISABLE;

if (HAL\_CAN\_Init(&hcan1) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN CAN1\_Init 2 \*/

/\*##-2- Configure the CAN Filter ###########################################\*/

sFilterConfig.FilterBank = 0;

sFilterConfig.FilterMode = CAN\_FILTERMODE\_IDMASK;

sFilterConfig.FilterScale = CAN\_FILTERSCALE\_32BIT;

sFilterConfig.FilterIdHigh = 0x0000;

sFilterConfig.FilterIdLow = 0x0000;

sFilterConfig.FilterMaskIdHigh = 0x0000;

sFilterConfig.FilterMaskIdLow = 0x0000;

sFilterConfig.FilterFIFOAssignment = CAN\_RX\_FIFO0;

sFilterConfig.FilterActivation = ENABLE;

sFilterConfig.SlaveStartFilterBank = 14;

if (HAL\_CAN\_ConfigFilter(&hcan1, &sFilterConfig) != HAL\_OK)

{

/\* Filter configuration Error \*/

Error\_Handler();

}

/\*##-3- Start the CAN peripheral ###########################################\*/

if (HAL\_CAN\_Start(&hcan1) != HAL\_OK)

{

/\* Start Error \*/

Error\_Handler();

}

/\*##-4- Activate CAN RX notification #######################################\*/

if (HAL\_CAN\_ActivateNotification(&hcan1, CAN\_IT\_RX\_FIFO0\_MSG\_PENDING) != HAL\_OK)

{

/\* Notification Error \*/

Error\_Handler();

}

/\*##-5- Configure Transmission process #####################################\*/

TxHeader.StdId = 0x001;

//TxHeader.ExtId = 0x01;

TxHeader.RTR = CAN\_RTR\_DATA;

TxHeader.IDE = CAN\_ID\_STD;

TxHeader.DLC = 8;

TxHeader.TransmitGlobalTime = DISABLE;

/\* USER CODE END CAN1\_Init 2 \*/

}

/\*\*

\* @brief I2C1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_I2C1\_Init(void)

{

/\* USER CODE BEGIN I2C1\_Init 0 \*/

/\* USER CODE END I2C1\_Init 0 \*/

/\* USER CODE BEGIN I2C1\_Init 1 \*/

/\* USER CODE END I2C1\_Init 1 \*/

hi2c1.Instance = I2C1;

hi2c1.Init.ClockSpeed = 100000;

hi2c1.Init.DutyCycle = I2C\_DUTYCYCLE\_2;

hi2c1.Init.OwnAddress1 = 0;

hi2c1.Init.AddressingMode = I2C\_ADDRESSINGMODE\_7BIT;

hi2c1.Init.DualAddressMode = I2C\_DUALADDRESS\_DISABLE;

hi2c1.Init.OwnAddress2 = 0;

hi2c1.Init.GeneralCallMode = I2C\_GENERALCALL\_DISABLE;

hi2c1.Init.NoStretchMode = I2C\_NOSTRETCH\_DISABLE;

if (HAL\_I2C\_Init(&hi2c1) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN I2C1\_Init 2 \*/

/\* USER CODE END I2C1\_Init 2 \*/

}

/\*\*

\* @brief I2S3 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_I2S3\_Init(void)

{

/\* USER CODE BEGIN I2S3\_Init 0 \*/

/\* USER CODE END I2S3\_Init 0 \*/

/\* USER CODE BEGIN I2S3\_Init 1 \*/

/\* USER CODE END I2S3\_Init 1 \*/

hi2s3.Instance = SPI3;

hi2s3.Init.Mode = I2S\_MODE\_MASTER\_TX;

hi2s3.Init.Standard = I2S\_STANDARD\_PHILIPS;

hi2s3.Init.DataFormat = I2S\_DATAFORMAT\_16B;

hi2s3.Init.MCLKOutput = I2S\_MCLKOUTPUT\_ENABLE;

hi2s3.Init.AudioFreq = I2S\_AUDIOFREQ\_96K;

hi2s3.Init.CPOL = I2S\_CPOL\_LOW;

hi2s3.Init.ClockSource = I2S\_CLOCK\_PLL;

hi2s3.Init.FullDuplexMode = I2S\_FULLDUPLEXMODE\_DISABLE;

if (HAL\_I2S\_Init(&hi2s3) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN I2S3\_Init 2 \*/

/\* USER CODE END I2S3\_Init 2 \*/

}

/\*\*

\* @brief SPI1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_SPI1\_Init(void)

{

/\* USER CODE BEGIN SPI1\_Init 0 \*/

/\* USER CODE END SPI1\_Init 0 \*/

/\* USER CODE BEGIN SPI1\_Init 1 \*/

/\* USER CODE END SPI1\_Init 1 \*/

/\* SPI1 parameter configuration\*/

hspi1.Instance = SPI1;

hspi1.Init.Mode = SPI\_MODE\_MASTER;

hspi1.Init.Direction = SPI\_DIRECTION\_2LINES;

hspi1.Init.DataSize = SPI\_DATASIZE\_8BIT;

hspi1.Init.CLKPolarity = SPI\_POLARITY\_LOW;

hspi1.Init.CLKPhase = SPI\_PHASE\_1EDGE;

hspi1.Init.NSS = SPI\_NSS\_SOFT;

hspi1.Init.BaudRatePrescaler = SPI\_BAUDRATEPRESCALER\_2;

hspi1.Init.FirstBit = SPI\_FIRSTBIT\_MSB;

hspi1.Init.TIMode = SPI\_TIMODE\_DISABLE;

hspi1.Init.CRCCalculation = SPI\_CRCCALCULATION\_DISABLE;

hspi1.Init.CRCPolynomial = 10;

if (HAL\_SPI\_Init(&hspi1) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN SPI1\_Init 2 \*/

/\* USER CODE END SPI1\_Init 2 \*/

}

/\*\*

\* @brief TIM9 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_TIM9\_Init(void)

{

/\* USER CODE BEGIN TIM9\_Init 0 \*/

/\* USER CODE END TIM9\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

/\* USER CODE BEGIN TIM9\_Init 1 \*/

/\* USER CODE END TIM9\_Init 1 \*/

htim9.Instance = TIM9;

htim9.Init.Prescaler = 84;

htim9.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim9.Init.Period = 0xFFFF;

htim9.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim9.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

if (HAL\_TIM\_Base\_Init(&htim9) != HAL\_OK)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

if (HAL\_TIM\_ConfigClockSource(&htim9, &sClockSourceConfig) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM9\_Init 2 \*/

/\* USER CODE END TIM9\_Init 2 \*/

}

/\*\*

\* @brief USART2 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_USART2\_UART\_Init(void)

{

/\* USER CODE BEGIN USART2\_Init 0 \*/

/\* USER CODE END USART2\_Init 0 \*/

/\* USER CODE BEGIN USART2\_Init 1 \*/

/\* USER CODE END USART2\_Init 1 \*/

huart2.Instance = USART2;

huart2.Init.BaudRate = 115200;

huart2.Init.WordLength = UART\_WORDLENGTH\_8B;

huart2.Init.StopBits = UART\_STOPBITS\_1;

huart2.Init.Parity = UART\_PARITY\_NONE;

huart2.Init.Mode = UART\_MODE\_TX\_RX;

huart2.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart2.Init.OverSampling = UART\_OVERSAMPLING\_16;

if (HAL\_UART\_Init(&huart2) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN USART2\_Init 2 \*/

/\* USER CODE END USART2\_Init 2 \*/

}

/\*\*

\* @brief USART3 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_USART3\_UART\_Init(void)

{

/\* USER CODE BEGIN USART3\_Init 0 \*/

/\* USER CODE END USART3\_Init 0 \*/

/\* USER CODE BEGIN USART3\_Init 1 \*/

/\* USER CODE END USART3\_Init 1 \*/

huart3.Instance = USART3;

huart3.Init.BaudRate = 115200;

huart3.Init.WordLength = UART\_WORDLENGTH\_8B;

huart3.Init.StopBits = UART\_STOPBITS\_1;

huart3.Init.Parity = UART\_PARITY\_NONE;

huart3.Init.Mode = UART\_MODE\_TX\_RX;

huart3.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart3.Init.OverSampling = UART\_OVERSAMPLING\_16;

if (HAL\_UART\_Init(&huart3) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN USART3\_Init 2 \*/

/\* USER CODE END USART3\_Init 2 \*/

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOE\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(CS\_I2C\_SPI\_GPIO\_Port, CS\_I2C\_SPI\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(OTG\_FS\_PowerSwitchOn\_GPIO\_Port, OTG\_FS\_PowerSwitchOn\_Pin, GPIO\_PIN\_SET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOD, LD4\_Pin|LD3\_Pin|LD5\_Pin|LD6\_Pin

|Audio\_RST\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin : CS\_I2C\_SPI\_Pin \*/

GPIO\_InitStruct.Pin = CS\_I2C\_SPI\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(CS\_I2C\_SPI\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : OTG\_FS\_PowerSwitchOn\_Pin \*/

GPIO\_InitStruct.Pin = OTG\_FS\_PowerSwitchOn\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(OTG\_FS\_PowerSwitchOn\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : PDM\_OUT\_Pin \*/

GPIO\_InitStruct.Pin = PDM\_OUT\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

GPIO\_InitStruct.Alternate = GPIO\_AF5\_SPI2;

HAL\_GPIO\_Init(PDM\_OUT\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : B1\_Pin \*/

GPIO\_InitStruct.Pin = B1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_EVT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(B1\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : BOOT1\_Pin \*/

GPIO\_InitStruct.Pin = BOOT1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(BOOT1\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : CLK\_IN\_Pin \*/

GPIO\_InitStruct.Pin = CLK\_IN\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

GPIO\_InitStruct.Alternate = GPIO\_AF5\_SPI2;

HAL\_GPIO\_Init(CLK\_IN\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : LD4\_Pin LD3\_Pin LD5\_Pin LD6\_Pin

Audio\_RST\_Pin \*/

GPIO\_InitStruct.Pin = LD4\_Pin|LD3\_Pin|LD5\_Pin|LD6\_Pin

|Audio\_RST\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

/\*Configure GPIO pin : OTG\_FS\_OverCurrent\_Pin \*/

GPIO\_InitStruct.Pin = OTG\_FS\_OverCurrent\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(OTG\_FS\_OverCurrent\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : MEMS\_INT2\_Pin \*/

GPIO\_InitStruct.Pin = MEMS\_INT2\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_EVT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(MEMS\_INT2\_GPIO\_Port, &GPIO\_InitStruct);

}

/\* USER CODE BEGIN 4 \*/

void HAL\_CAN\_RxFifo0MsgPendingCallback(CAN\_HandleTypeDef \*hcan)

{

/\* Get RX message \*/

if (HAL\_CAN\_GetRxMessage(hcan, CAN\_RX\_FIFO0, &RxHeader, RxData) != HAL\_OK)

{

/\* Reception Error \*/

Error\_Handler();

}

TxData[0] = RxData[0];

TxData[1] = RxData[1];

if (HAL\_CAN\_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailbox) != HAL\_OK)

{

/\* Transmission request Error \*/

Error\_Handler();

}

}

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

\_\_disable\_irq();

while (1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

#endif /\* USE\_FULL\_ASSERT \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/

**<raspberry pi의 can\_ubidot.c 파일>**

#include <stdio.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <string.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include <fcntl.h>

#include <sys/ioctl.h>

#include <net/if.h>

#include <linux/can.h>

#include <linux/can/raw.h>

#include <signal.h>

#include <pthread.h>

#define MAXLINE 1024

#define UBIDOTS\_SERVER "50.23.124.68"

#define UBIDOTS\_PORT 80

#define UBIDOTS\_TOKEN "BBFF-EoCMqDLcyV34ATeqPXOcvuugxbdMLY"

#define UBIDOTS\_DEVICE "raspberry\_pi"

int soc;

int read\_can\_port;

int dev\_pid=0;

struct can\_frame frame\_rx;

char temp[10];

char humid[10];

int open\_port(const char \*port)

{

struct ifreq ifr;

struct sockaddr\_can addr;

/\* open socket \*/

soc = socket(PF\_CAN, SOCK\_RAW, CAN\_RAW);

if(soc < 0)

{

return (-1);

}

addr.can\_family = AF\_CAN;

strcpy(ifr.ifr\_name, port);

if (ioctl(soc, SIOCGIFINDEX, &ifr) < 0)

{

return (-1);

}

addr.can\_ifindex = ifr.ifr\_ifindex;

fcntl(soc, F\_SETFL, O\_NONBLOCK);

if (bind(soc, (struct sockaddr \*)&addr, sizeof(addr)) < 0)

{

return (-1);

}

return 0;

}

int send\_port(struct can\_frame \*frame)

{

int retval;

retval = write(soc, frame, sizeof(struct can\_frame));

if (retval != sizeof(struct can\_frame))

{

return (-1);

}

else

{

return (0);

}

}

/\* run in a thread \*/

void\* read\_port(void\* data)

{

int recvbytes = 0;

read\_can\_port = 1;

while(read\_can\_port)

{

struct timeval timeout = {1, 0};

fd\_set readSet;

FD\_ZERO(&readSet);

FD\_SET(soc, &readSet);

if (select((soc + 1), &readSet, NULL, NULL, &timeout) >= 0)

{

if (!read\_can\_port)

{

break;

}

if (FD\_ISSET(soc, &readSet))

{

recvbytes = read(soc, &frame\_rx, sizeof(struct can\_frame));

if(recvbytes)

{

printf("dlc = %d, id = 0x%x, temp = %d%d, humidity = %d%d\n",

frame\_rx.can\_dlc, frame\_rx.can\_id, frame\_rx.data[0], frame\_rx.data[1],frame\_rx.data[2],frame\_rx.data[3]);

sprintf(temp, "%d.%d", frame\_rx.data[0], frame\_rx.data[1]);

sprintf(humid, "%d.%d", frame\_rx.data[2],frame\_rx.data[3]);

}

}

}

usleep(1000);

}

}

int main(int argc, char \*\*argv) {

open\_port("can0");

pthread\_t candata;

pthread\_create(&candata,NULL,read\_port,NULL);

struct sockaddr\_in serveraddr;

int server\_sockfd;

int client\_len;

char buf[MAXLINE];

char message[64];

if((server\_sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) == -1) {

perror("error : ");

return 1;

}

serveraddr.sin\_family = PF\_INET;

serveraddr.sin\_addr.s\_addr = inet\_addr(UBIDOTS\_SERVER);

serveraddr.sin\_port = htons(80);

client\_len = sizeof(serveraddr);

if(connect(server\_sockfd, (struct sockaddr\*)&serveraddr, client\_len) == -1) {

perror("connect error : ");

return 1;

}

while (1)

{

//can data print

printf( "temp:%s, humid:%s%\n\n", temp, humid);

//send to ubidots

memset(buf, 0x00, MAXLINE);

sprintf(message, "{\"temperature\": %s, \"humidity\": %s}", temp, humid);

sprintf(buf, "POST /api/v1.6/devices/%s/?token=%s HTTP/1.1\r\nHost: things.ubidots.com\r\nContent-Type: application/json\r\nContent-Length: %d\r\n\r\n%s", UBIDOTS\_DEVICE, UBIDOTS\_TOKEN, (int)strlen(message), message);

if(write(server\_sockfd, buf, strlen(buf)) <= 0) {

perror("write error : ");

return 1;

}

memset(buf, 0x00, MAXLINE);

if(read(server\_sockfd, buf, MAXLINE) <= 0) {

perror("read error: ");

return 1;

}

printf("server:%s\n", buf);

sleep(10);

}

close(server\_sockfd);

pthread\_join(candata,NULL);

return 0;

}