**[Z6] Wybrana platforma – UART**

|  |  |  |
| --- | --- | --- |
| **Grupa laboratoryjna:** L11  **Podgrupa: 4** | Paweł Kolec (155 873) |  |
| Adam Nowacki (155 838) |
| **Prowadzący zajęcia:** | **dr inż. Ariel Antonowicz** |

**ZDJĘCIA ZREALIZOWANEGO UKŁADU**

|  |
| --- |
|  |
|  |

**KOD ZREALIZOWANEGO ALGORYTMU - Arduino**

|  |
| --- |
| #include <OneWire.h>  #include <DS18B20.h>  #define ONEWIRE\_PIN 19  // Pin do komunikacji 1-Wire  float mala[18] = {0.0};  // Tablica przechowująca ostatnie 18 odczytów temperatury  short int numer = 0;     // Indeks do zapisu kolejnych odczytów  // Adres czujnika  byte address[8] = {0x28, 0x8, 0xC, 0x79, 0x97, 0x2, 0x3, 0x84};  OneWire onewire(ONEWIRE\_PIN);  DS18B20 sensors(&onewire);  // Funkcja obliczająca najmniejszą wartość w tablicy  float mini(float mala[18]) {      float a = 1000000.0;      for (int i = 0; i < 18; i++) {          if (mala[i] < a) a = mala[i];      }      return a;  }  // Funkcja obliczająca największą wartość w tablicy  float maxi(float mala[18]) {      float a = -273.15;      for (int i = 0; i < 18; i++) {          if (mala[i] > a) a = mala[i];      }      return a;  }  // Funkcja obliczająca średnią wartość bez minimum i maksimum  float srednia(float mala[18]) {      float suma = 0;      for (int i = 0; i < 18; i++) {          suma += mala[i];      }      suma = (suma - mini(mala) - maxi(mala)) / 16;      return suma;  }  void setup() {      Serial.begin(9600);      sensors.begin();      sensors.request(address);  }  void loop() {      if (sensors.available()) {          float temperature = sensors.readTemperature(address);  // Odczyt temperatury          mala[numer] = temperature;  // Zapis do tablicy          numer++;          if (numer == 18) {              float avg\_temperature = srednia(mala);  // Obliczenie średniej              Serial.println(avg\_temperature);  // Wyświetlenie wyniku              numer = 0;          }          sensors.request(address);  // Żądanie kolejnego pomiaru          delay(1);      }  } |

**KOD ZREALIZOWANEGO ALGORYTMU – Wybrana platforma**

|  |
| --- |
| import Adafruit\_BBIO.UART as UART import serial import sqlite3 import time  *# Inicjalizacja UART* def setup\_uart():  UART.setup("UART1") *# Ustawienie portu UART1* ser = serial.Serial(port="/dev/ttyS1", baudrate=9600, timeout=1) *# Konfiguracja połączenia UART* return ser *# Zwraca obiekt serial do dalszego użytku  # Tworzenie bazy danych (jeśli nie istnieje)* def create\_db():  conn = sqlite3.connect('temperatura.db') *# Połączenie z bazą SQLite* c = conn.cursor()  c.execute('''CREATE TABLE IF NOT EXISTS temperatura  (id INTEGER PRIMARY KEY AUTOINCREMENT,  MEASURED\_AT DATETIME DEFAULT CURRENT\_TIMESTAMP,  temperatura REAL)''') *# Tworzenie tabeli do przechowywania temperatur* conn.commit()  conn.close()  *# Wstawianie średniej temperatury do bazy danych* def insert\_db(avg\_temp):  conn = sqlite3.connect('temperatura.db') *# Połączenie z bazą* c = conn.cursor()  c.execute("INSERT INTO temperatura (temperatura) VALUES (?)", (avg\_temp,)) *# Wstawienie wartości* conn.commit()  conn.close()  *# Wyświetlanie danych z bazy* def show\_db():  conn = sqlite3.connect('temperatura.db') *# Połączenie z bazą* c = conn.cursor()  c.execute("SELECT \* FROM temperatura") *# Pobranie wszystkich rekordów* measurements = c.fetchall()  for measurement in measurements:  print(f"ID: {measurement[0]} Znacznik czasowy: {measurement[1]} Temperatura: {measurement[2]} C") *# Wyświetlenie rekordów* conn.close()  *# Funkcja do odczytu danych z UART i przetwarzania* def read\_uart\_data(ser):  if ser.isOpen(): *# Sprawdzenie, czy port UART jest otwarty* message = ser.readline().decode("utf-8").strip() *# Odczyt i oczyszczenie danych* return message  return None *# Zwraca None, jeśli port nie jest otwarty  # Główna funkcja programu* def main():  ser = setup\_uart() *# Inicjalizacja UART* create\_db() *# Tworzenie bazy danych* while True:  message = read\_uart\_data(ser) *# Odczyt danych z UART* if message:  try:  avg\_temp = float(message)   print(f"Odczytana temperatura: {avg\_temp} C")   insert\_db(avg\_temp) *# Zapis temperatury do bazy danych* show\_db() *# Wyświetlenie zawartości bazy* except ValueError:  print("Nieprawidłowy format danych temperatury")   time.sleep(1) *# Przerwa między odczytami* if \_\_name\_\_ == "\_\_main\_\_":  main() |