Data via LoRaWAN

DuPe (www.dupedup.cz)
Prototype Version: 11/2024-1

Purpose:

Measure data by senzor. Data is periodicaly sent it via LoRaWAN network to data server at provider LoRaWAN. Received data is POSTed to data web server into CSV file. Data is readed by WEB browser and displayed as graph.

Hardware components:

1) MCU Cube Cell HTTC AB01 + Antenna 868 MHz (Heltec Automation)

- Master Chip ASR6501 (48 MHz ARM® Cortex® M0+ MCU)
- Memory 128KB internal FLASH; 16KB internal SRAM
- LoRa Chipset SX1262
- Frequency 470~510 MHz, 863~923 MHz
- Max TX Power 22dB ± 1dB
- Receiving sensitivity -135 dBm
- UART x 1; SPI x 1; I2C x 1; SWD x 1; 12-bits ADC input x 1; 8–channel DMA engine; GPIO x 8
- Micro USB x 1; LoRa Antenna interface(IPEX) x 1;
- USB to Serial Chip CP2102
- Battery 3.7V Lithium(SH1.25 x 2 socket)
- Solar Energy 5.5～7V solar panel
- Low Power Deep Sleep 3.5μA
- Operating temperature -20 ~ 70 ℃
- Dimensions 41.5 x 25 x 7.6 mm

2) SENSOR DIGITAL TEMPERATURE DS18B20

- Power supply range: 3.0V to 5.5V
- Operating temperature range: -55°C to +125°C (-67F to +257F)
- Storage temperature range: -55°CC to +125°C (-67F to +257F)
- Accuracy over the range of -10°C to +85°C: ±0.5°

SENSOR DIGITAL HUMIDITY, PRESSURE AND TEMPERATURE BME280

- Digital interface I^2C (up to 3.4 MHz)
- Supply voltage VDD main supply voltage range: 1.71 V to 3.6 V
- Current consumption 1.8 μA @ 1 Hz humidity and temperature
- 2.8 μA @ 1 Hz pressure and temperature
- 3.6 μA @ 1 Hz humidity, pressure and temperature
- 0.1 μA in sleep mode
- Operating range -40...+85 °C, 0...100 % rel. humidity, 300...1100 hPa
- Package 2.5 mm x 2.5 mm x 0.93 mm metal lid LGA

3) Solar Panel 6V

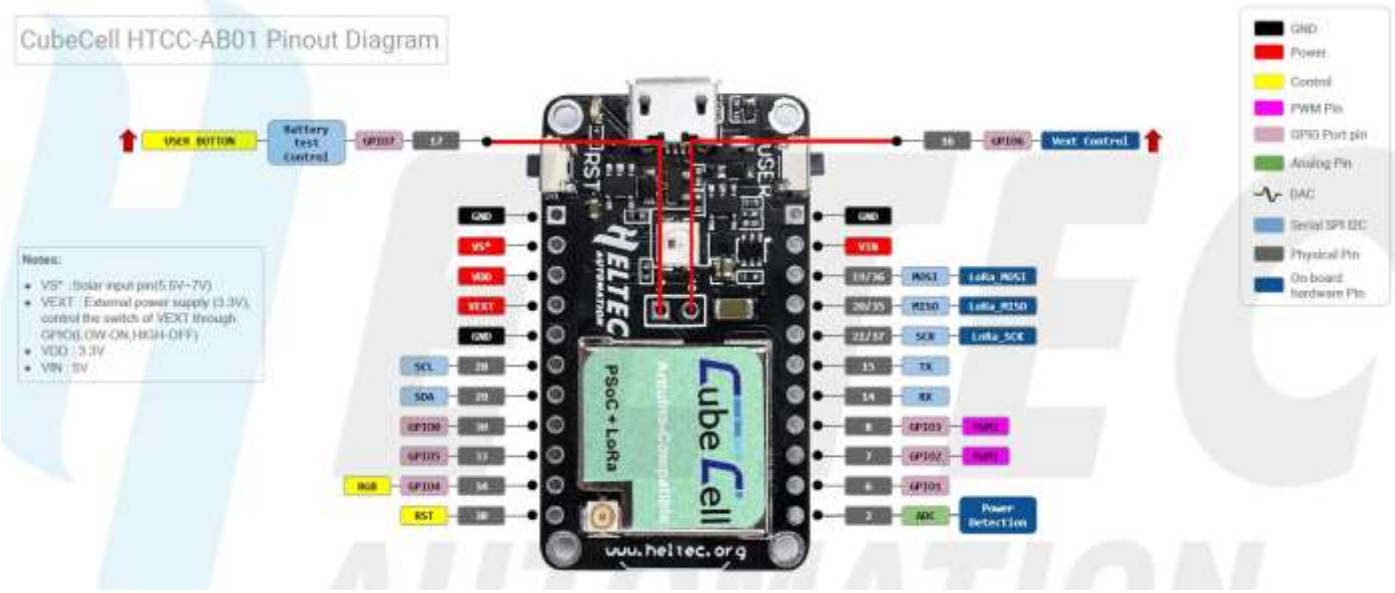
- Output min 300mA (2W)
- Size 110x136x3mm / 52g

4) LiPol 1S 900mAh 3.7V

Circuit:

Cube Cell Modul (PIN 5V tolerant)	**SENZOR DS18B20** (One Wire)	**SENZOR BME280** (I2C)	**Other**
28		SCL	
29	DATA	SDA	
VDD (3.3 V OUT)	VCC	VCC	
VS (5.5 - 7V)			Solar Panel 6V

Battery Connector			ACU LiPo 4V
GND	GND	GND	GND ACU/Solar



Processing:

TEMPERATURE/HUMIDITY/PRESSURE - > SENSOR -> SENSOR REGISTRY -> SENSOR BUS (I2C) -> MCU DATA BUS (I2C) -> MCU (STM32) -> MCU DATA BUS -> OLED (I2C) -> LORA RADIO chip (SPI) -> RADIO 868MHz -> LORAWAN GW (UP TO 10 km) -> LORAWAN SERVER REST API -> DATA POST -> WWW SERVER (PHP CODE) -> DATA FILE
DATA FILE -> HTML/JS CODE (API GOOGLE CHARTS) - > WWW USER BROWSER

Data : 10 - 20 B
Cycle: 300s
LoRa: 868 MHz , SF 12, BW 125, DR about 300 bps
LoRaWAN: OTTA parameters (devEUI, apEUI, appKEY) were generated on The Things Network
LoRaWAN Provider: CeskeRadiokomunikace Datové centrum, přenos dat, Cloud a IoT | České Radiokomunikace

Software Components:

- C/C++ VSCode / Platform IO + Heltec Support
- https://github.com/HelTecAutomation/CubeCell-Arduino
- https://github.com/milesburton/Arduino-Temperature-Control-Library
- https://github.com/adafruit/Adafruit_BME280_Library

Code:

platformio.ini

```
[env:cubecell_board]
platform = heltec-cubecell
board = cubecell_board
framework = arduino
monitor_speed = 115200
build_flags =
   -D ACTIVE_REGION=LORAMAC_REGION_EU868
```

```
    -D REGION_EU868
    -D LORAWAN_CLASS=CLASS_A
    -D LORAWAN_DEVEUI_AUTO=0
    -D LORAWAN_NETMODE=true
    -D LORAWAN_ADR=true
    -D LORAWAN_UPLINKMODE=true
    -D LORAWAN_Net_Reserve=false
    -D LORAWAN_AT_SUPPORT=0
    -D LORAWAN_PREAMBLE_LENGTH=8
    -D LORAWAN_DebugLevel=2
    -D RGB=1
lib_deps = milesburton/DallasTemperature@^4.0.4
```

setup-lora.h

```
#define AT_SUPPORT  0
#define RGB 1
#define DELAYTIME 1000
#define ACTIVE_REGION LORAMAC_REGION_EU868
#define CLASS_MODE CLASS_A
#define LORAWAN_DEVEUI_AUTO 0
#define CubeCell_BoardPlus
#define REGION_EU868
#define LORAWAN_NETMODE true
#define LORAWAN_ADR true
#define LORAWAN_UPLINKMODE true
#define LORAWAN_Net_Reserve false
#define LORAWAN_AT_SUPPORT 0
#define LORAWAN_PREAMBLE_LENGTH 8
#define LORAWAN_DebugLevel 2

/* OTAA para*/
uint8_t devEui[] = { };
uint8_t appEui[] = { };
uint8_t appKey[] = { };

/* ABP para*/
uint8_t nwkSKey[] = { };
uint8_t appSKey[] = { };
uint32_t devAddr =  ( uint32_t )0x007e6ae1;
/*LoraWan channelsmask, default channels 0-7*/
uint16_t userChannelsMask[6]={ 0x00FF,0x0000,0x0000,0x0000,0x0000,0x0000 };
/*LoraWan region, select in arduino IDE tools*/
LoRaMacRegion_t loraWanRegion = ACTIVE_REGION;
/*LoraWan Class, Class A and Class C are supported*/
DeviceClass_t  loraWanClass = LORAWAN_CLASS;
/*the application data transmission duty cycle.  value in [ms].*/
uint32_t appTxDutyCycle = 300000;
/*OTAA or ABP*/
bool overTheAirActivation = LORAWAN_NETMODE;
/*ADR enable*/
bool loraWanAdr = LORAWAN_ADR;
/* set LORAWAN_Net_Reserve ON, the node could save the network info to flash, when node reset not need to join
again */
bool keepNet = LORAWAN_NET_RESERVE;
/* Indicates if the node is sending confirmed or unconfirmed messages */
bool isTxConfirmed = LORAWAN_UPLINKMODE;
uint8_t appPort = 2;
uint8_t confirmedNbTrials = 4;
```

MCU Source Code

```
// DUPE 12/2024
```

```
// HW HELTEC LORA CUBECELL AB01, LORAWAN CRO
// CO https://heltec-automation-docs.readthedocs.io/en/latest/index.html
#include <Arduino.h>
#include "LoRaWan_APP.h"
#include <OneWire.h>
#include "DallasTemperature.h"
#include "setup-lora.h"
#define ONE_WIRE_BUS SDA
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensor(&oneWire);
int count = 0;

static void prepareTxFrame( uint8_t port )
{
 pinMode(Vext, OUTPUT);
  count++;
  digitalWrite(Vext, LOW);
  delay(1000);
  uint vbat = getBatteryVoltage();
  sensor.requestTemperatures(); // Send the command to get temperatures
  Serial.println("DONE");
  // After we got the temperatures, we can print them here.
  // We use the function ByIndex, and as an example get the temperature from the first sensor only.
  float tempC = sensor.getTempCByIndex(0);
  // Check if reading was successful
  if (tempC != DEVICE_DISCONNECTED_C)
  {
   Serial.print("Temperature: ");
   Serial.println(tempC);
  }
  else
  {
   Serial.println("Error: Could not read temperature data");
  }
  appDataSize = 10;
  char payload[10];
  String payloads = "";
  int temp = (tempC+40)*100;
  payloads.concat(temp);
  payloads.concat(vbat);
  payloads.toCharArray(payload,10);
  // appData = (unsigned char *)payload;
  strcpy ((char*)appData, payload);
  Serial.print("Message=");
  Serial.println(payloads);
}

void setup() {
        boardInitMcu();
        Serial.begin(115200);
#if(AT_SUPPORT)
        enableAt();
#endif
        deviceState = DEVICE_STATE_INIT;
        LoRaWAN.ifskipjoin();
}
```

```
void loop()
{
	switch( deviceState )
	{
		case DEVICE_STATE_INIT:
		{
#if(AT_SUPPORT)
			getDevParam();
#endif
			printDevParam();
			LoRaWAN.init(loraWanClass,loraWanRegion);
			deviceState = DEVICE_STATE_JOIN;
			break;
		}
		case DEVICE_STATE_JOIN:
		{
			LoRaWAN.join();
			break;
		}
		case DEVICE_STATE_SEND:
		{
			prepareTxFrame( appPort );
			LoRaWAN.send();
			deviceState = DEVICE_STATE_CYCLE;
			break;
		}
		case DEVICE_STATE_CYCLE:
		{
			// Schedule next packet transmission
			txDutyCycleTime = appTxDutyCycle + randr( 0, APP_TX_DUTYCYCLE_RND );
			LoRaWAN.cycle(txDutyCycleTime);
			deviceState = DEVICE_STATE_SLEEP;
			break;
		}
		case DEVICE_STATE_SLEEP:
		{
			LoRaWAN.sleep();
			break;
		}
		default:
		{
			deviceState = DEVICE_STATE_INIT;
			break;
		}
	}
}
```