**Česká zemědělská univerzita v Praze**

**Technická fakulta**

**Katedra elektrotechniky a automatizace**

# Diplomová práce

## Systém pro sledování stavu včelstva

**Radim Vilčko**

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Technická fakulta

# ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Radim Vilčko

Informační a řídicí technika v agropotravinářském komplexu

Název práce

**Systém pro sledování stavu včelstva**

Název anglicky

**Hive health monitoring system**

---

**Cíle práce**

Cílem práce je vytvoření inteligentního sledovacího systému stavu včelstva. Systém bude umožňovat sledovat vnitřní a vnější prostředí včelstva, bude se jednat např. o teploty, vlhkosti, zvuk, vibrace, hmotnost včelstva, světelné podmínky, srážky, rychlost a směr větru a kameru. V práci budou navrženy možnosti akčního zásahu pro změnu vnitřních podmínek včelstva. Zařízení bude umožňovat zasílat informace pomocí SMS či emailem a provádět automatická opatření či uživatelskou interakci se systémem.

**Metodika**

Prostudování hardwarových a softwarových možností řešení. Navržení několika variant provedení úlohy. Výběr nejvhodnější varianty s kritickým hodnocením návrhu. Specifikace funkcí modelu podle cílů práce.

**Doporučený rozsah práce**

60stran, bez příloh

**Klíčová slova**

Arduino, detektor, WiFi, smart, automatizace

---

**Doporučené zdroje informací**

DENNIS, Andrew K. *Raspberry Pi home automation with Arduino : automate your home with a set of exciting projects for the Raspberry Pi!.* Birmingham: Packt Publishing, 2013. ISBN 978-1-78439-920-7.

GOODWIN, Steven. *Smart home automation with Linux and raspberry Pi.* New York: Apress, 2013. ISBN 978-1-4302-5887-2.

Learn IoT Programming Using Node-RED, Begin to Code Full Stack IoT Apps and Edge Devices with Raspberry Pi, NodeJS, and Grafana. BPB Publications, 2022. ISBN 9391392385.

MORRISS, S. Brian. *Automated manufacturing systems : actuators, controls, sensors, and robotics.* New York: Glencoe, 1995. ISBN 0028023315.

---

**Předběžný termín obhajoby**

2023/2024 LS – TF

**Vedoucí práce**

doc. Ing. Miloslav Linda, Ph.D.

**Garantující pracoviště**

Katedra elektrotechniky a automatizace

Elektronicky schváleno dne 3. 2. 2023

**doc. Ing. Monika Hromasová, Ph.D.**

Vedoucí katedry

Elektronicky schváleno dne 8. 3. 2023

**doc. Ing. Jiří Mašek, Ph.D.**

Děkan

V Praze dne 01. 11. 2023

**Čestné prohlášení**

Prohlašuji, že svou diplomovou práci "Systém pro sledování stavu včelstva" jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne datum odevzdání _____31.3.2024_____

**Poděkování**

Rád bych touto cestou poděkoval panu docentu Miloslavu Lindovi za odborné vedení při psaní této práce. Za to že mi věnoval svůj čas, energii a korigoval moje směřování při psaní této diplomové práce. Dále děkuji kolegovi ze zaměstnání Zbyňku Piškulovi za pomoc při 3D tisku potřebných součástek. A rovněž svým dětem že mi poskytli trochu toho volného času.

# Systém pro sledování stavu včelstva

**Abstrakt**

Cílem této Diplomové práce byla konstrukce zařízení, které nám bude monitorovat aktuální stav včelstva. Bylo zvoleno monitorování teploty jak venkovní, tak vnitřní. Rovněž použitý senzor měřil také vzdušnou vlhkost. Jako další byla nejzásadnější konstrukce váhy pro měření přírůstků medu. To je pro včelaře velice důležitá informace. Veškerá data byla v reálném čase ukládána do Arduino Cloudu. Historicky je na nich možno vidět jednotlivé trendy pomocí jednak webové aplikace, tak aplikace pro mobilní zařízení. Posledním prvkem, který poskytoval informace bylo GSM. To primárně může sloužit k evidenci a monitoringu polohy jednotlivých úlů a zároveň jako bezpečnostní prvek, kdyby byl by včelín ukraden.

Řešení byla postaveno na mikrokontroleru Arduino MKR 1010 WiFi a k němu doplněny jednotlivé senzory a další podpůrné prvky jako GSM Shield, solární panel s příslušnou elektronikou pro úpravu napětí, kabeláží, baterií a konstrukcí rámu váhy. Vše muselo fungovat jako ostrovní systém bez možnosti připojení ke stálé energetické síti.

**Klíčová slova:** WiFi, IoT, Arduino, DHT22, GSM, Cloud, MKR 1010, Úl, Váha

# Bee colony monitoring system

**Abstract**

The aim of this thesis was to design a device that will monitor the current status of the bee colony. Both outdoor and indoor temperature monitoring was chosen. Also, the sensor used also measured the humidity in the air. The most important next was the design of a scale to measure the honey increments. This is very important information for beekeepers. All data was stored in real time in the Arduino Cloud. Historically, individual trends can be viewed using both a web app and a mobile app. The last element that provided information was GSM. This primarily can be used to record and monitor the location of individual hives and also as a security feature should a hive be stolen.

The solution was built on an Arduino MKR 1010 WiFi microcontroller and added individual sensors and other supporting elements such as a GSM Shield, a solar panel with appropriate electronics for voltage adjustment, wiring, batteries and a scale frame structure. Everything had to work as an island system without the possibility of connection to a permanent power grid.

**Keywords**: WiFi, IoT, Arduino, DHT22, GSM, Cloud, MKR 1010, Hive, Scale, Hive

# Obsah

# Seznam obrázků

# Seznam tabulek



# Seznam diagramů

# Seznam použitých zkratek

GPS - Global Positioning System

WiFi - Wireless Fidelity

A/D (AD) - Analog-to-Digital

DC - Direct Current

AC - Alternating Current

LoRa - Long Range

GSM - Global System for Mobile Communications

3G – 3 Generation

LTE - Long Term Evolution

AES - Advanced Encryption Standard

IoT - Internet of Things

MS - Mobile Station

BTS - Base Transceiver Station

BSC - Base Station Controller

SGSN - Serving GPRS Support Node

GGSN - Gateway GPRS Support Node

HSCDS - High Speed Circuit Switched Data

GPRS - General Packet Radio Service

8-PSK - Quadrature Phase Shift Keying

EDGE - Enhanced Data Rates for GSM Evolution

LTE-NB IoT - Long Term Evolution - Narrowband Internet of Things

UE - User Equipment

E-UTRAN - The Evolved UMTS Terrestrial Radio Access Network

eNodeB - evolved Node B

QoS - Quality of Service

EPC - The Evolved Packet Core

eDRX - Discontinuous Reception

PSM - Power Saving Mode

LTE-M - Machine Type Communication

WPA3 WiFi - Protected Access 3

ECDH - Elliptic Curve Diffie-Hellman

SHA-254 - Secure Hash Algorithm 256-bit

I2C - Inter-Integrated Circuit

GPIO - General Purpose Input/Output

PWM - Pulse-Width Modulation

SDA - Serial Data Line

SCL - Serial Clock Line

SCLK - Serial Clock

MOSI - Master Out Slave In

MISO - Master In Slave Out

SS - Slave Select

GND – Ground

VIN - Voltage Input

NTC - Negative Temperature Coefficient

IDE - Integrated Development Environment

# 1 Úvod

V současné době, kdy se celý svět potýká s výzvami spojenými s udržitelností životního prostředí a ztrátou biologické diverzity, se stává otázka ochrany a monitoringu včelstev, které hrají klíčovou roli v ekosystémech díky opylování, stále důležitější. S ohledem na tyto výzvy byla předložená diplomová práce zaměřena na konstrukci a realizaci inovativního zařízení pro monitorování aktuálního stavu včelstev. Cílem této práce bylo nejen poskytnout včelařům důležité informace o teplotě, vlhkosti a přírůstcích medu v jejich úlech, ale také nabídnout řešení, které by mohlo přispět k lepší péči o včelí kolonie a zvýšení jejich produkce.

Význam monitorování včelstev je nesporný, neboť teplota a vlhkost v úlu mohou mít přímý vliv na zdraví a produktivitu včel. Příliš vysoká nebo nízká teplota, stejně jako nadměrná vlhkost, mohou vést k nemocem včelích kolonií, snížení produkce medu a dokonce k úhynu celých kolonií. Včasné detekce těchto nepříznivých podmínek umožňuje včelařům zasáhnout a předejít možným problémům. Kromě teploty a vlhkosti je sledování hmotnosti úlu zásadní pro určení přírůstků medu, což je pro včelaře klíčová informace nejen z ekonomického hlediska, ale i pro správné načasování sklizně medu.

Řešení představené v této práci bylo založeno na využití mikrokontroleru Arduino MKR 1010 WiFi, doplněného o specificky vybrané senzory pro měření teploty, vlhkosti a hmotnosti, a GSM Shield pro komunikaci a lokalizační služby. Kromě toho byla pro autonomní provoz zařízení v terénu navržena solární napájecí jednotka spolu s baterií a elektronikou pro úpravu napětí. Veškerá získaná data byla v reálném čase ukládána do Arduino Cloudu, což umožňovalo jejich snadný přístup a analýzu prostřednictvím webové aplikace nebo mobilního zařízení. Toto řešení nejenže nabízí včelařům přehled o stavu jejich včelstev, ale také zvyšuje bezpečnost úlů díky monitorování jejich polohy a potenciálnímu varování v případě krádeže.

Tato práce představuje komplexní přístup k monitorování včelstev, který integruje moderní technologie a poskytuje užitečné nástroje pro řízení a ochranu včelích kolonií. V následujících kapitolách je podrobně rozebrána konstrukce zařízení, výběr komponent, programování a integrace systému, jakož i analýza získaných dat a jejich význam pro praktické včelaření.

# 2 Cíl práce

Cílem je sestavit funkční zařízení, které nám bude sloužit jako monitorovací systém. Pomocí něj budeme sbírat data o teplotě, vlhkosti, hmotnosti a polohových GPS souřadnic.

- V první fázi se zaměříme na vhodný výběr součástek. Jednat se bude zejména o senzory na měření výše zmíněných veličin.
- Dále zhodnotíme jednotlivé desky Arduino z rodiny MKR a zvolíme pro daný projekt tu nejvhodnější.
- Zkompletování funkčního řešení. To znamená hardwarové sestavení funkčního celku připojení senzorů pomocí sběrnic k mikrokontroleru.
- Zvolit vhodné napájení. Protože se počítá s tím, že se jedná o ostrovní systém bez možnosti použití přímého napájení ze sítě. Bude použito napájení ze solárního panelu s tím, že přebytky budou ukládány do lokálního akumulátoru.
- Sestavení vhodného kódu ve vývojovém prostředí. Pomocí něj budeme celý proces sběru a odesílání dat řídit.
- Zajistit vhodné úložiště pro naměřená data. Možnost jejich zobrazení za použití webového prohlížeče.
- Požadavek zasílání notifikací na základě námi nastavených mezních hodnot. Notifikace budou odesílány jednak emailem tak pomocí aplikace v mobilním zařízení.
- Na závěr zhodnotíme náročnost projektu a jak po stránce hardwarové, tak co se psaní zdrojového kódu týče. A samozřejmě zhodnotíme i jeho finanční stránku.

# 3 Metodika

Pro naplnění stanovených cílů použijeme vybrané součástky viz. bod číslo jedna. V dalších částech této práce je zhodnotíme a popíšeme jejich technické specifikace a vlastnosti.

- Výběr a zakomponování senzorů a čidel pro sbírání dat:
  - Hmotnost (tenzometr)
  - Teplota a vlhkost (DHT22 senzor)
  - Polohové informace z GPS (Arduino MKR GPS shield)
- Fyzické připojení těchto senzorů k vybrané vývojové desce Arduino MKR 1010 WiFi.
- Připojení Arduino desky do Arduino Cloud prostředí.
- Sestavení programu v Arduino MKR 1010, který bude celý systém řídit. Sběr a odesílání dat do Arduino Cloudu. To zahrnuje vhodnou volbu jednotlivých knihoven pro daná zařízení.
- V prostředí Arduino Cloud vyrobit web, který bude naměřené hodnoty interpretovat pomocí widgetů.
- Nastavení zaslání notifikací pomocí emailu a aplikace v mobilním telefonu.

Na závěr se podíváme i na časovou a finanční náročnost celého projektu. Budeme hodnotit ne jenom cenu zakoupených součástek ale i například cenu za tarif, pokud bude použito spojení pomocí operátora, který poskytuje přenosové prostředí.

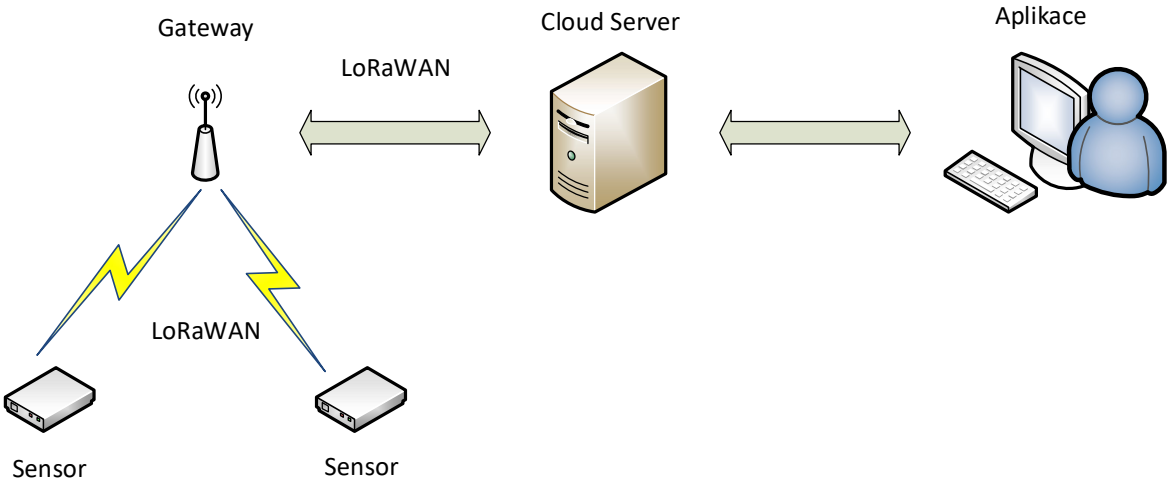
# 4 Teoretická východiska

Zde se podrobněji zaměříme na jednotlivé dílčí prvky našeho návrhu. Budeme se zde zabývat jednotlivými deskami. Z hlediska jejich konektivity do internetu. Popíšeme vlastnosti a zamyslíme se nad tím podle jakých kritérií můžeme volit. Dále zde budou podrobněji popsány jednotlivé prvky jako jsou. DHT22 senzor teploty a vlhkosti, HX 711 24 bit A/D převodník, tenzometr, solárního panel, DC-DC XL6009 napěťový převodník a v neposlední řadě Arduino MKR GPS Shield.

## 4.1 Vývojová deska Arduino

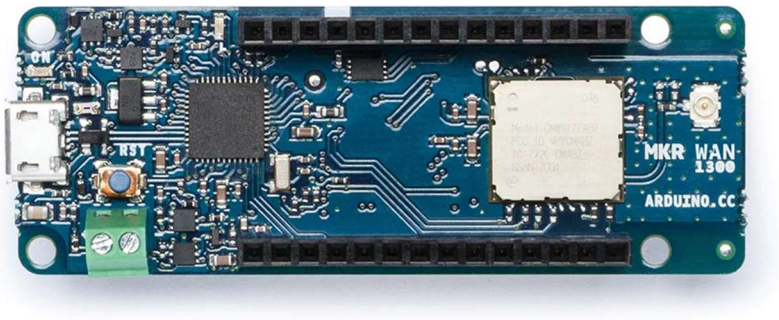
Jak již bylo zmíněno výše srdcem tohoto projektu a celé diplomové práce je deska Arduino MKR. Řada Arduino MKR alespoň v čase, kdy byla tato diplomová práce psána existovala pro sítě LoRa, GSM/3G, narrowband LTE a pro kratší vzdálenosti jsou to pak zejména Bluetooth a WiFi sítě. Každé řešení má svoje vlastnosti a volíme jej s ohledem na typ aplikace, kterou na nich budeme provozovat.

### 4.1.1 Síť LoRa (Long Range)




*Obrázek 1: Topologie sítě LoRaWAN*

Se sítí LoRa je kompatibilní Arduino MKR WAN 1300 nebo 1310. Tyto desky jsou vybaveny modulem Murata CMWX1ZZABZ, který podporuje LoRaWAN (Long Range Wide Area Network) bezdrátový protokol. Jeho podporované frekvence jsou 868 MHz a 915 MHz. V našich zeměpisných šířkách požíváme frekvence v rozsahu 863-870 MHz. [1]




*Obrázek 2: Arduino MKR WAN 1300 [1]*

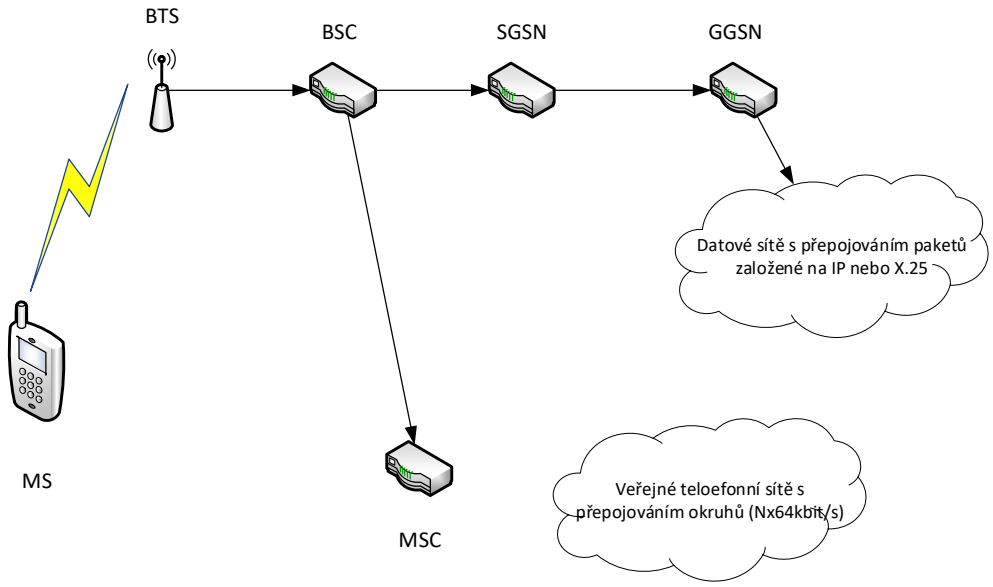
LoRa je bezdrátová síť určená především pro internet věcí (IoT). Jako každá technologie má své výhody a nevýhody. Jeho hlavní výhodu spatřuji v energetické nenáročnosti na koncové zařízení, dlouhý dosah řádově desítky kilometrů v otevřeném terénu, jednotky kilometrů ve městě a vysokou odolnost proti rušení. [2] [3]

Bezpečnost přenášených dat je zajištěna AES symetrickou šifrou (Advanced Encryption Standard) o délce klíče 128 bitů. Sítě LoRaWAN šifrují provoz hned na několika úrovních. Network Session Key (NwkKey) pro přenos mezi koncovým zařízením a serverem a dále klíč pro šifrování aplikačních dat takzvaný Application Session Key (AppSKey). [2] [4]

Počítá se s tím, že toto zařízení bude napájeno po dubu několika let bez nutnosti výměny baterie. Z toho však také plyne „nevýhoda“ lépe spíš vlastnost, a to jsou malé přenosové rychlosti. Řádově zde hovoříme o jednotkách maximálně desítkách kilobitů za sekundu (od 0,3 kbit/s do 50 kbit/s). Také časový interval je mnohem delší. Čím delší je tím delší je i životnost baterie. Jsou aplikace kde nám stačí odeslat informaci v intervalu jednou za 24 hodin a méně. Typicky se zde nabízí například odečty energií v domácnostech. [2] [3]

### 4.1.2 Síť GSM/3G

Další technologií, jak je možno se připojit je pomocí mobilního operátora, který podporuje sítě 2 nebo 3 generace. Zde se omezím na popis pouze sítě 2 generace, a to z důvodů že sítě 3 generace byli v ČR kompletně vypnuty všemi třemi operátory v roce 2023. Uvolněné frekvence budou použity pro sítě LTE a 5G. 5G síť by měla uživatelům nabídnout teoreticky rychlosti okolo 1000 Mbit/s [5]




*Obrázek 3: Přenos dat prostřednictvím EDGE [6]*

Starší sítě 2 generace budou podporovány nejméně do roku 2028. [7] Složení jednotlivých komponent v síti je následovné.

MS (Mobile Station) jedná se o koncové zařízení, v našem případě by se jednalo o Arduino MKR GSM 1400. Procesor je stejný jako v předchozím případě a v podstatě celá struktura je stejná. V čem je tato deska jiná od předchozí a dalších které zde budou zmíněny je koprocesor starající se o komunikaci GMS/3G. Ten je od společnosti U-blox a jedná se o SARA-U201, nízko příkonovou čipovou sadou pracující v různých pásmech mobilního rozsahu (GSM 850 MHz, E-GSM 1900 MHz, DCS 1800 MHz, PCS 1900 MHz). [8]

Další specifikace řady MKR její periferie a specifikace, které jsou stejné budou rozebrány v samostatné kapitole této práce.


*Obrázek 4: MKR GSM 1400 [8]*

Dále je zde BTS (Base Transceiver Station) jedná se o klíčovou jednotku vybavenou anténami zajišťující radiovou komunikaci s koncovým zařízením. Následuje BSC (Base Station Controller). Ten nám zajišťuje správu a řízení několika BTS. Dále prvek SGSN (Serving GPRS Support Node), který komunikuje s radiovou částí. Pro přenos do jiných datových sítí např. do internetu je implementován datový uzel GGSN (Gateway GPRS Support Node) [6]

Síť 2 generace si prošla za dobu svého používání postupnou evolucí, hlavně co se týkalo použitých přenosových rychlostí. Kdy se v začátcích dosahovalo rychlostí 9,6 kbit/s postupně se přešlo na rychlosti 14,4 kbit/s pro datový přenos. S přechodem na tzv. 2,5 generaci HSCDS (High Speed Circuit Switched Data) bylo dosahováno rychlostí 115 kbit/s dále potom při použití technologie GPRS (General Packet Radio Service) 171 kbit/s zde se jedná již o paketově orientovaný přenos, který je výhodnější pro přenos dat. Další a poslední navýšení přenosových rychlostí v této generaci bylo dosaženo použitím 8-PSK (Quadrature Phase Shift Keying) ta je použita prostřednictvím tzv. EDGE (Enhanced Data Rates for GSM Evolution), kde bylo teoreticky možné dosáhnou rychlostí 473,6 kbit/s [6]

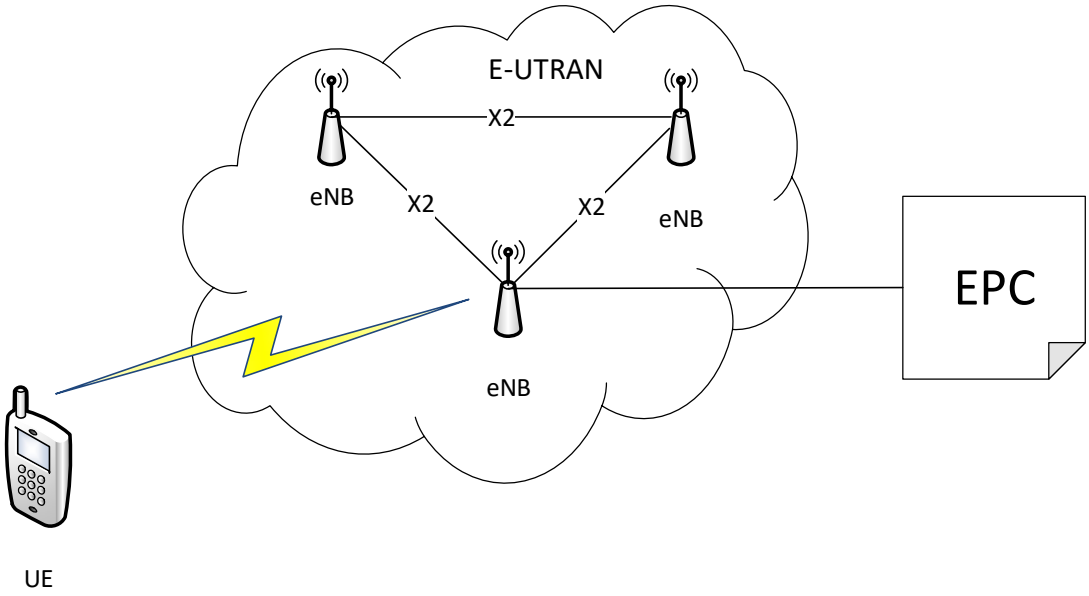
Nicméně ani tento druh připojení nebudu požívat z důvodů zmíněných výše, a to je pomalu morální zastaralost a postupný útlum těchto sítí. Perspektivnější se je jeví LTE-NB a sítě páté generace.

### 4.1.3 Síť LTE-NB

V současnosti jsou nejpoužívanější sítě typu LTE-NB a mají zdaleka nejlepší pokrytí území ČR. Pro tyto sítě je určen MKR NB 1500. Tato deska z rodiny MKR, co se osazení týče je shodná s předchozími pouze se liší v modulu pro bezdrátovou komunikaci u-blox SARA-R410M-02B. [9]


*Obrázek 5: Arduino NB 1500 [9]*

Síťová architektura se skládá z následujících součástí. User Equipment (UE) je koncové zařízení v našem případě Arduino NB 1500. The Evolved UMTS Terrestrial Radio Access Network (E-UTRAN) jedná se pozemní přístupovou radiovou síť. Tato síť obsahuje Základnové Stanice (eNodeB) ty jsou zodpovědné za správu spektra, řízení QoS a poskytování konektivity pro jednotlivá zařízení. Tyto eNodeB jsou propojeny s The Evolved Packet Core (EPC) jedná se o vnitřní core síť. Tyto dva prvky spolu bývají propojeny metalickými, optickými nebo radiovými spoji. Pro nás je důležité že EPC nám poskytuje konektivitu do internetu. [10]



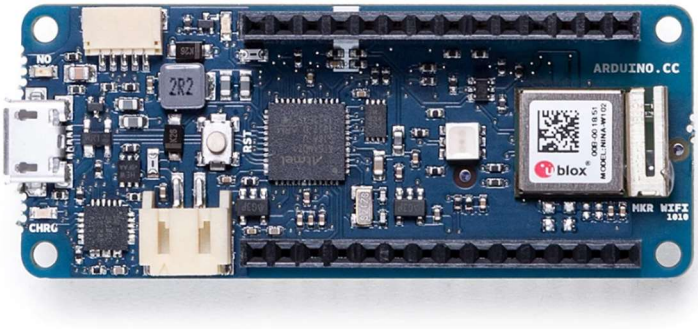

*Obrázek 6: LTE Síťová architektura [10]*

Dále je zde nutno zmínit že existují dva různé LTE standardy pro IoT. První z nich je NB-IoT vyznačuje se úzkou šířkou přenášeného pásma 200 kHz to zlepšuje jeho průnik signálu a má lepší pokrytí. Velmi vysoká energetická účinnost s tím spojená dlouhá životnost zařízení na jednu baterii. Přenosové rychlosti do 250 kbit/s. [11]

Druhá je Cat-M1 (LTE-M) má větší šířku přenášeného pásma 1,4 MHz a podporuje duplexní přenos o rychlosti 1 Mb/s. Nižší latence dále podporuje hlasové hovory. Má větší nároky na režii a šířku pásma. Proto nedosahuje takové životnosti jako předešlá NB-IoT. Jak NB-IoT tak i Cat-M1 mohou prodloužit svoji výdrž díky Discontinuous Reception (eDRX) což znamená přerušení příjmu a Power Saving Mode (PSM) úsporný režim. Tím výrazně dokáže snížit spotřebu energie. Obě řešení mají svá opodstatnění a bude záležet na typu aplikace pro který je budeme chtít použít. [11]

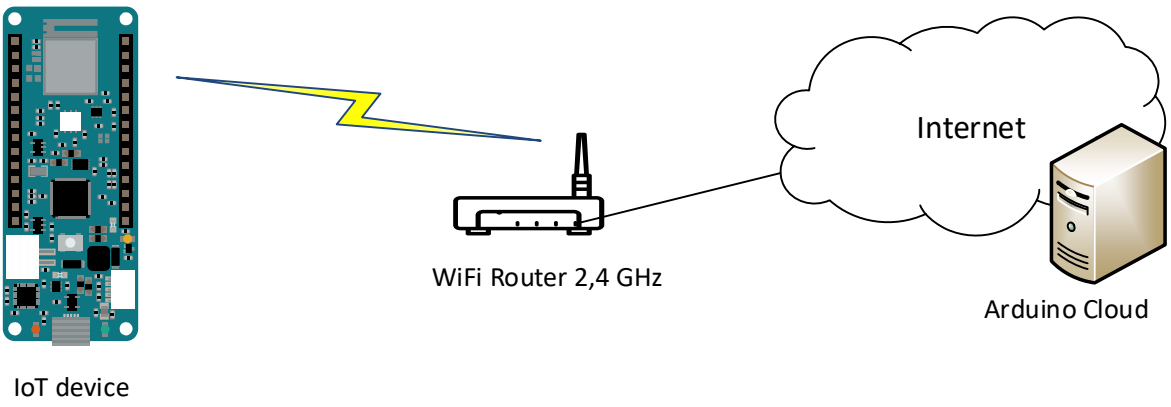
### 4.1.4 WiFi MKR-WiFi 1010

Deska, která nám umožňuje připojení do této sítě pomocí Wifi je Arduino MKR WiFi 1010. Tato deska je bude použita pro projekt monitorování stavu včelstva. [12]




*Obrázek 7: Arduino MKR 1010 [12]*

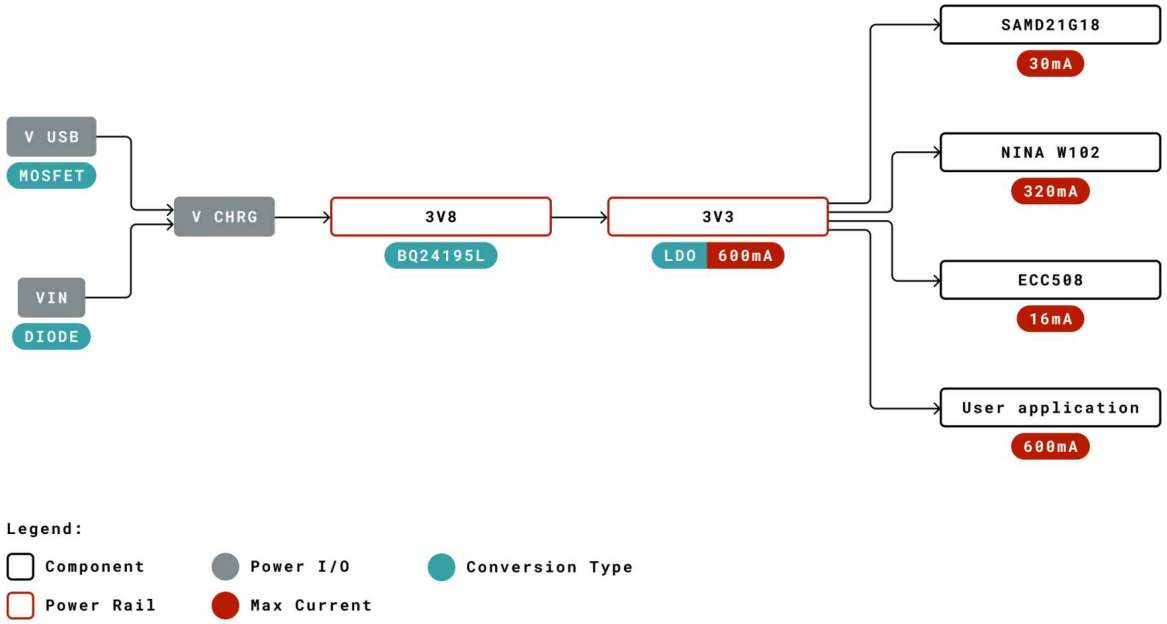
Procesor je opět 32bitový procesor Arm® Cortex®-M0 32-bit SAMD21 běžící na frekvenci 48 MHz stejně jako u ostatních desek rodiny MKR. Konektivitu nám zajišťuje modul od společnosti u-blox Nina-W102, tato čipová sada s nízkou spotřebou energie pracuje v pásmu 2,4 GHz podporující standardy IEEE 802.11 b/g/n. Zároveň podporuje WPA3 Wi-Fi (Protected Access III). Ten obsahuje minimálně AES 128 symetrický šifrovací algoritmus. Dále bezpečnost je zajištěna prostřednictvím kryptografického čipu Microchip® ECC508. Tento koprocesor je určen například k uchovávání klíčů, certifikátů nebo dat. Dále podporuje ECDH (Elliptic Curve Diffie-Hellman) jedná se o protokol, kterým si koncové zařízení bezpečně vymění klíče. Posledním důležitým úkolem je podpora SHA-254 (Secure Hash Algorithm 256-bit) což je hashovací funkce. Pomocí této funkce můžeme ukládat například hesla nebo kontrolovat integritu přenášených dat. [12]




*Obrázek 8: WiFi připojení do sítě*

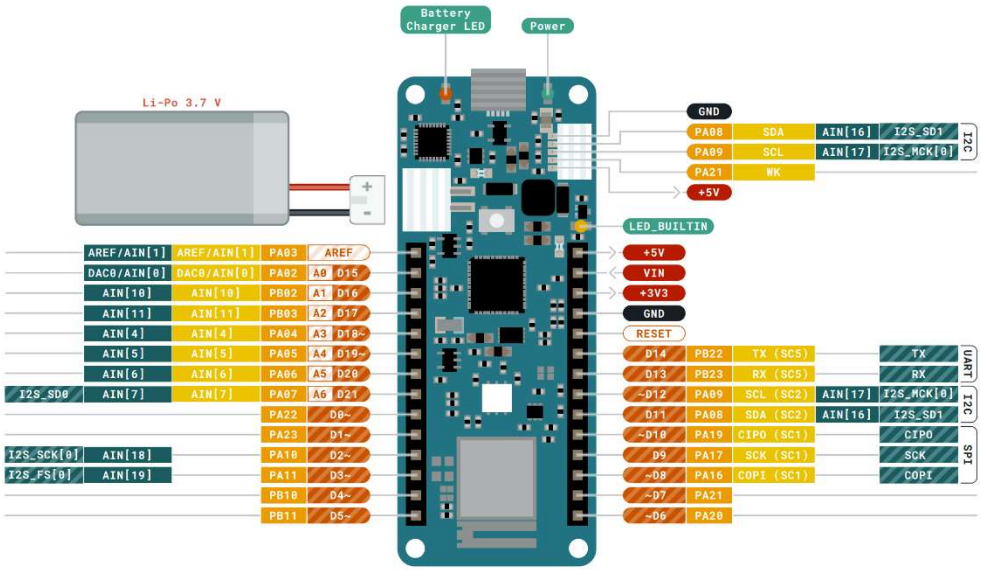
#### 4.1.4.1 Power Tree MKR WiFi 1010

Pro správné dimenzování zátěže tohoto mikrokontroleru je zde vložen power tree. Ten zobrazuje, jakým způsobem je přerozdělena proudová zátěž pro jednotlivé mikroprocesory, koprocesory a kolik zbývá pro uživatelské aplikace. Z tohoto jasně vyplívá že pro naši aplikaci máme k dispozici 600 mA.




*Obrázek 9: Power Tree Arduino MKR WiFi 1010 [13]*

#### 4.1.4.2 Periferie MKR 1010 WiFi.




*Obrázek 10: Pinout diagram MKR WiFi 1010 [14]*

Obrázek číslo 10 nám detailně popisuje jakým způsobem jsou na desce vyvedeny jednotlivé vstupně výstupní piny a další periferie. To je pro větší přehled prezentováno i v následující tabulce.

*Tabulka 1: Piny MKR WiFi 1010*

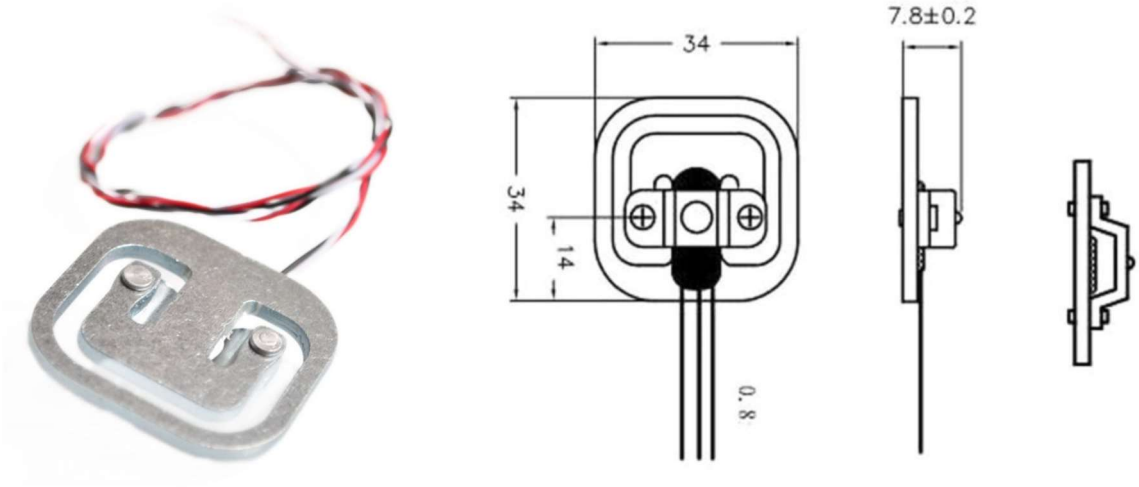
| Pin | Funkce | Typ | Popis |
|---|---|---|---|
| 1 | AREF | Analog | Analog Reference. |
| 2 | A0/DAC0 | Analog | ADC in/DAC out, Lze použít jako as GPIO |
| 3 | A1 | Analog | ADC in, Lze použít jako as GPIO |
| 4 | A2 | Analog | ADC in, Lze použít jako as GPIO |
| 5 | A3 | Analog | ADC in, Lze použít jako as GPIO |
| 6 | A4/SDA | Analog | ADC in, I2C SDA, Lze použít jako as GPIO |
| 7 | A5/SCL | Analog | ADC in, I2C SCL, Lze použít jako as GPIO |
| 8 | A6 | Analog | ADC in, Lze použít jako as GPIO |
| 9 | D0 | Digital | GPIO, Lze použít jako as PWM |
| 10 | D1 | | GPIO, Lze použít jako as PWM 11 D2/PWM Digital GPIO, Lze použít jako as PWM |
| 12 | D3/PWM | Digital | GPIO, Lze použít jako as PWM |
| 13 | D4/PWM | Digital | GPIO, Lze použít jako as PWM |
| 14 | D5/PWM | Digital | GPIO, Lze použít jako as PWM |
| 15 | D6 | Digital | GPIO, Lze použít jako as PWM |
| 16 | D7 | Digital | GPIO Lze použít jako as PWM |
| 17 | D8/MOSI | Digital | SPI MOSI, Lze použít jako as GPIO, Lze použít jako as PWM |
| 18 | D9/SCK | Digital | SPI SCK, Lze použít jako as GPIO, Lze použít jako as PWM |
| 19 | D10/MISO | Digital | SPI MISO, Lze použít jako as GPIO |
| 20 | D11/SDA | Digital | I2C SDA, Lze použít jako as GPIO |
| 21 | D12/SCL | Digital | I2C SCL, Lze použít jako as GPIO |
| 22 | D13/RX | Digital | USART RX, Lze použít jako as GPIO |
| 23 | D14/TX | Digital | USART TX, Lze použít jako as GPIO |
| 24 | RESETN | Digital | Reset input |
| 25 | GND | Napájení | Zem 26 +3V3 Výstupní napětí |
| 27 | VIN | Napájení | Vstupní pin pro napájení desky +5V |
| 28 | +V5 | Napájení | |

## 4.2 Periferie a senzory

Po diskusi se včelařem vyplynuly jako nejužitečnější výstupy, které lze ze včelína získat zejména údaje o hmotnosti (respektive přírůstcích hmotnosti), vnitřní a vnější teplota, vlhkost a v případě že dotyčný obhospodařuje více včelínů údaje o poloze. Proto se bude tato práce věnovat právě sběru a interpretaci těchto získaných dat.

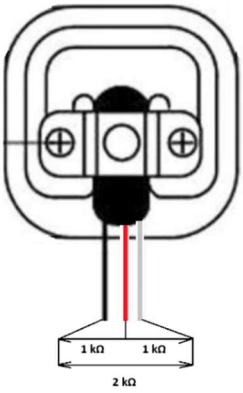
### 4.2.1 Váhový sensor 50 kg

Jedná se o váhový senzor tvořený tenzometrem. Jeho princip spočívá ve změně odporu při mechanickém namáhání. V zapojení budou použity celkem čtyři senzory. Každý z nich pracuje v rozsahu 0-50 kg. Jedná se o velmi malé změny odporu, proto je potřeba doplnit jej o operační zesilovač s AD převodníkem. Celé snímání této změny bude řešeno kompenzační metodou, takzvaným zapojením do Wheatstonova můstku.



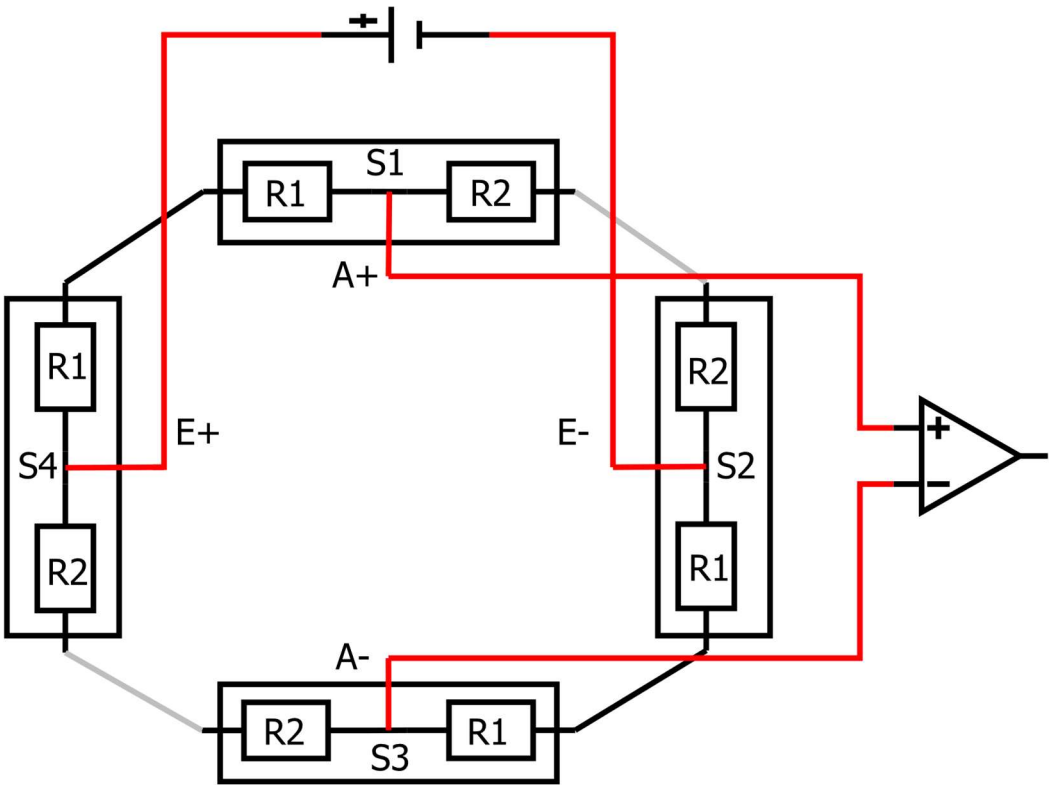

*Obrázek 11: Tenzometr 50Kg [15]*

Tento senzor má dva protilehlé tenzometry nalepené proti sobě. V případě že působíme tlakem ve směru shora dolů odpor se na jedné straně vlivem deformace zmenšuje a na druhé zvětšuje. Oba tenzometry mají v klidovém stavu hodnotu 1000 Ω jak je naznačeno na obrázku číslo 12.




*Obrázek 12: Vývody ze senzoru*

Na dalším obrázku je schematicky znázorněno, jakým způsobem jsou fyzicky propojeny barevné dráty jednotlivých senzorů. Přičemž S1 až S4 jsou jednotlivé senzory a R1 a R2 odpory v nich. Ty jsou propojeny tak aby tvořily Wheatstonův můstek. Reálně bude výstup vyveden do AD 24bitového převodníku HX711, který je ideální pro tyto aplikace. Bude popsán v následujících kapitolách. Zde je naznačeno výstupem ze svorek A+ a A-, které vedou do operačního zesilovače k dalšímu zpracování a vyhodnocení naměřených hodnot.




*Obrázek 13. Propojení tenzometrických senzorů*

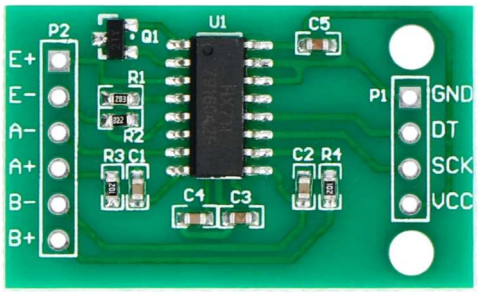
### 4.2.2 Princip fungování

Na začátku v ideálním případě je můstek dokonale vyvážen a na jeho obou vstupech máme stejné napětí. Protože vlivem zatížení dojde k mechanickému namáhání a tím pádem k prodloužení, respektive zkrácení vodičů dojde i ke změně odporů v těchto vodičích. Vše podle následujícího vztahu:

$$R = \rho * \frac{l}{S} \tag{4.1}$$

Kde R je odpor vodiče (Ω), ρ je měrný odpor materiálu v Ω·m (ohm-metry), l je dálka v metrech (m) a S je průřez vodiče ($m^2$). Tyto změny jsou příliš malé na to, aby byly změřeny klasickou metodou. Proto zde požíváme zapojení do můstku. Změna rovnovážného stavu je přivedena na vstup diferenciálního operačního zesilovače. Zde je diferenciální napětí zesíleno a dále obvodem HX711 převedeno na digitální hodnotu 24bitovou hodnotu.

### 4.2.3 HX711 24-Bit Analogově digitální převodník

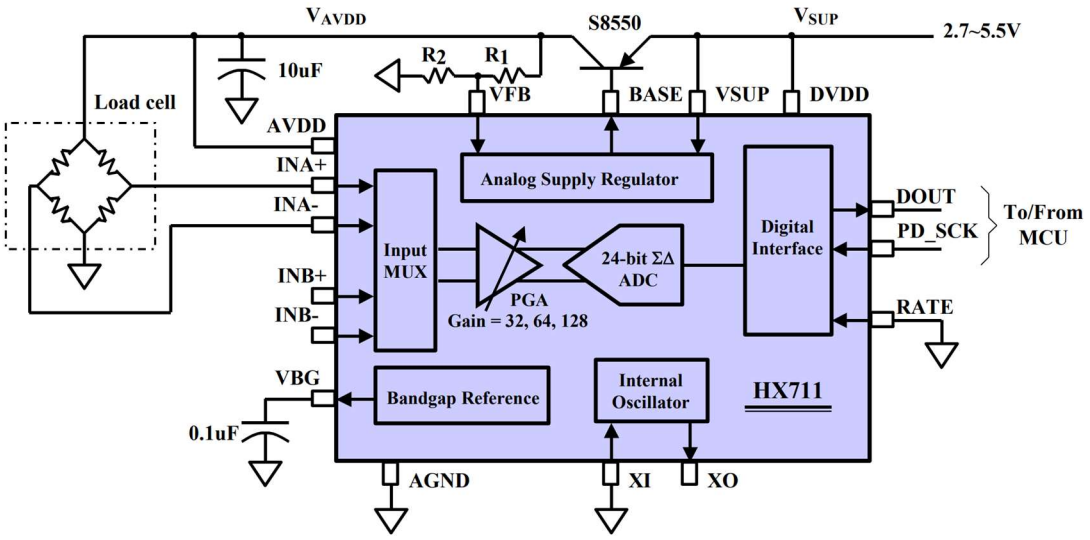



*Obrázek 14: AD převodník KX711*

V kombinaci se senzory viz. předchozí kapitola je použitý 24bitový AD převodník HX711. Toto patentované řešení firmy Avia Semiconductor je přímo navrženo pro použití při konstrukci snímače hmotnosti a další průmyslové aplikace, které jsou zapojeny do můstku. Tento převodník je vybaven dvěma kanály. Kanál A má volitelné zesílení 128 a 64. To odpovídá diferenciálnímu vstupu v plném rozsahu ±20 mV respektive ±40 mV, jestliže je k napájení AVDD připojeno napětí o velikosti 5 V. [16]

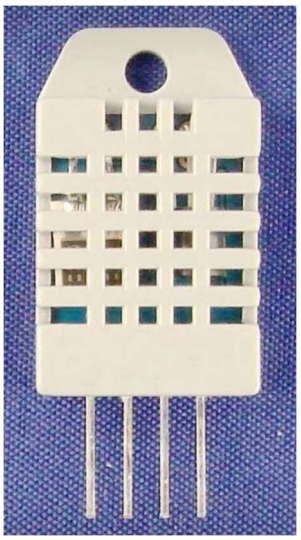
Kanál B má pevné zesílení 32 a jedná se diferenciální nízko šumový vstup. Vstup hodinového signálu je volitelný a může pocházet z externího zdroje nebo ze zdroje krystalu, který je součástí čipu. Tento převodník pracuje s napájecím napětím v rozsahu 2.6 až 5.5 V a při zátěži odebírá méně než 1.5 mA v klidovém režimu pak 1 μA. Dalším zajímavým parametrem je teplotní rozsah od -40 do +85 °C. [16]

Při použití oscilátoru na čipu pracuje převodník s rychlostí 10 SPS při hodnotě log 0 na pinu RATE, nebo 80 SPS při hodnotě log 1na témže pinu. [16]



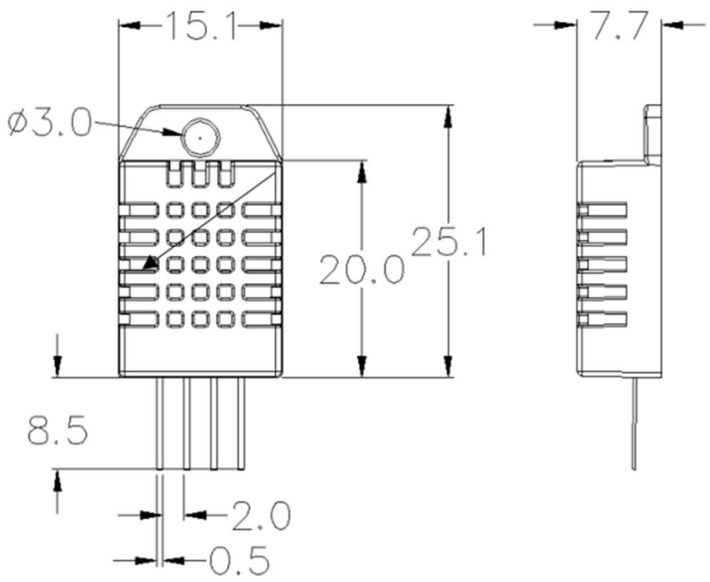

*Obrázek 15: Blokový diagram se zapojenou váhou [16]*

### 4.2.4 DHT22 senzor teploty a relativní vlhkosti



*Obrázek 16: DHT22 [17]*

Dalším použitým senzorem je DHT22 pro měření teploty v rozsahu -40 ~ +80 °C s přesností ±0,5 °C. Dále rozsah měřené relativní vlhkosti je od 0–100 % s přesnost ±2-5 %. Tento senzor při měří vzdušné vlhkost používá kapacitní senzorový prvek. Pro měření odporu je použitý NTC termistor. Napájecí napětí by mělo být v rozmezí 3.3–6 V stejnosměrných. Je-li v pracovním režimu odběr se pohybuje okolo 2.5 mA. [17] [18]




*Obrázek 17: Pinout a rozměry DHT22 [17]*

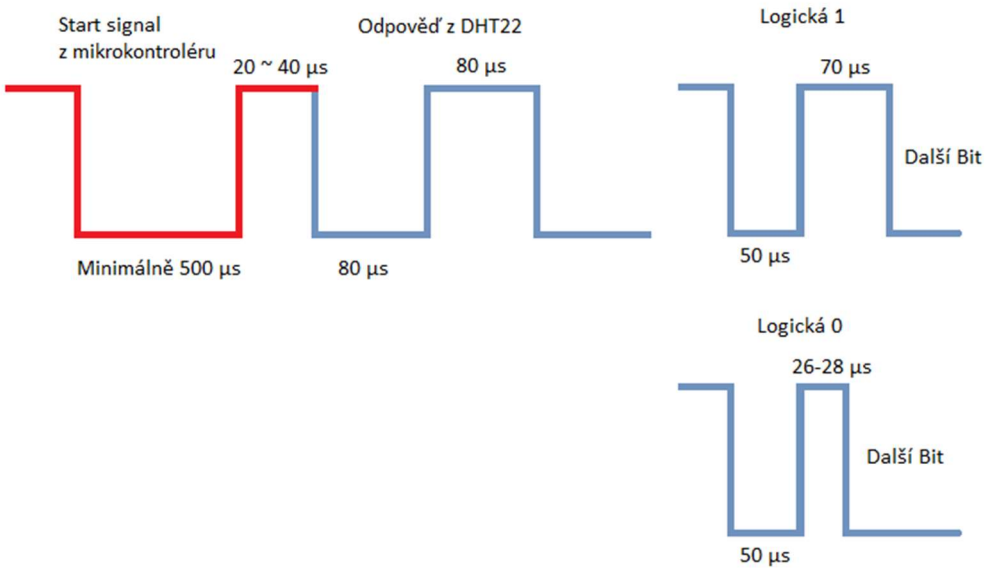
*Tabulka 2: Funkce pinů*

| Pin | Funkce |
|---|---|
| 1 | VDD napájení |
| 2 | Data signál |
| 3 | --- |
| 4 | GND referenční zem |

Pro komunikaci s mikrokontrolerem používá DHT22 jednu datovou sběrnici je to druhý pin z leva viz. obrázek číslo 17. Doporučená frekvence měření je minimálně 2 sekundy. [17]

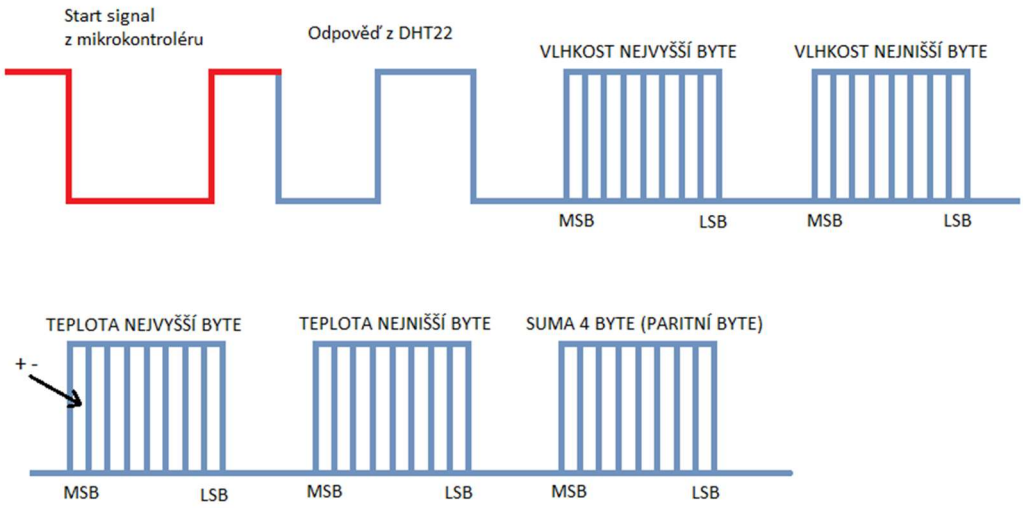
### 4.2.5 Komunikace DHT22 s mikrokontrolerem

Pro navázání komunikace s mikrokontrolerem a vyčtení hodnot po sběrnici pošle kontrolér startovací signál. Což je v případě tohoto senzoru nízká úroveň signálu po dobu 500 µs následovaná vysokou úrovní v intervalu 20-40 µs. Dále následuje komunikace po sběrnici od sensoru k mikrokontroleru. Kdy je logická 1 reprezentována délkou signálu 70 µs ve vysoké úrovni a před ní je 50 µs v nízké úrovni. Logická 0 začíná opět nízkou úrovní v intervalu 50 µs po jejím skončení následuje ve vysoké úrovni 28 µs. Vše je graficky znázorněno na obrázku 18. [17] [18]




*Obrázek 18: Časový průběh začátku komunikace a logických hodnot [18]*

DHT22 odesílá oba údaje ve dvou bajtech po osmi bitech. U teploty je-li první bit logická 1 je hodnota záporná v opačném případě to znamená že hodnota nabývá kladné hodnoty. Na konec je vyslán kontrolní součet všech bitů. [17] [18]




*Obrázek 19: Datová sériová komunikace DHT22 [18]*

Celkový čas nutný k přenosu kompletní informace do mikrokontroleru je cca 5 ms. Nicméně v dokumentaci je doporučeno počkat alespoň 2 sekundy mezi přenosy. [18]

### 4.2.6 Napájení

V napájecím řetězci bude použit solární panel jako zdroj dobíjení akumulátoru. Dále Boost-buck modul solárního napájení DC-DC XL6009. Ten slouží k úpravě vstupně, výstupního napětí ze solárního panelu. A jako poslední Lithium-polymerová baterie o kapacitě 4Ah.

#### 4.2.6.1 Solární panel

*Obrázek 20: Solární panel 6 V*

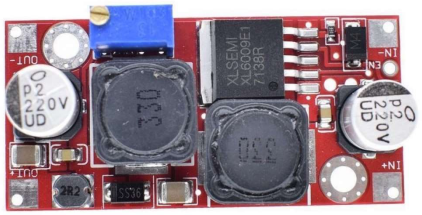
Parametry tohoto solárního panelu jsou:

- Výstupní napětí 6 V
- Výstupní výkon 3.3 W
- Výstupní proud 0–550 mA
- Hmotnost 82 g
- Typ panelu polykrystalický
- Rozměry 135 x 165 x 2,5 mm

[19]

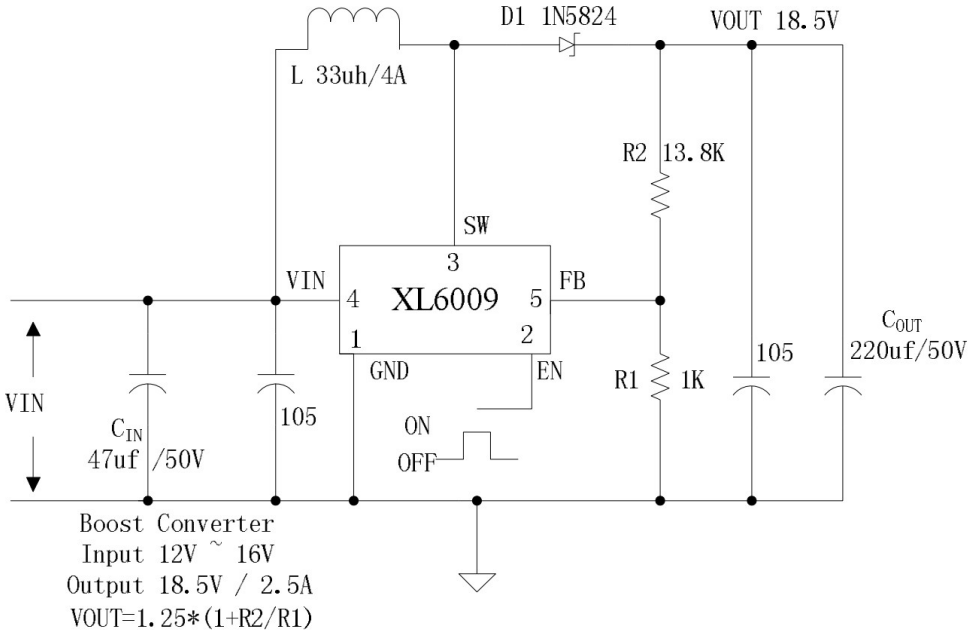
#### 4.2.6.2 Boost-buck step up/down modul DC-DC XL6009

Tento univerzální step up/down měnič požijeme jako stabilizovaný zdroj napětí. Pro nastavení vyššího napětí pracujeme v režimu boost. Jestliže chceme výstupní napětí ze směru vstup výstup snížit pracujeme v režimu buck.



*Obrázek 21: Boost-buck step DC-DC XL6009*

Základem této desky je modul XL6009, který celý proces zvyšování, respektive snižování úrovně vstupně výstupního napětí řídí. Na následujícím obrázku je náhradní schéma a zapojení pro boost režim. [20]



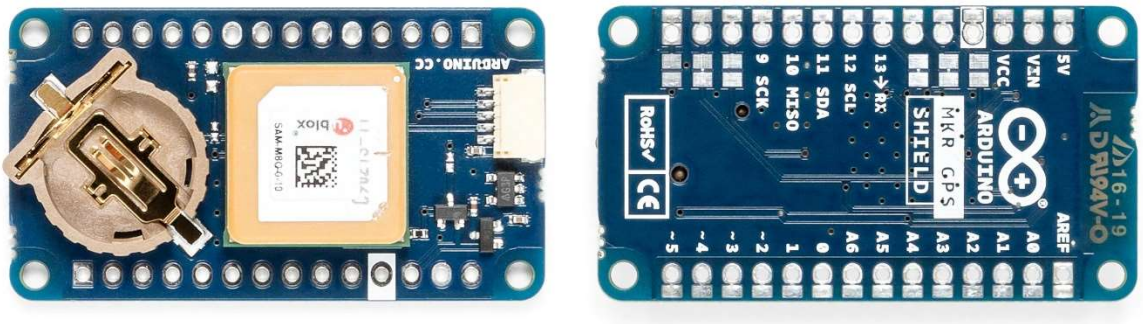

*Obrázek 22: Náhradní schéma zapojení XL6009 Boost-Buck režim*

Vstupní napětí, které chceme regulovat je zapojeno mezi piny 4 a 1. Pin 1 je zároveň spojen se zemí viz. přechozí schéma. Pin 2 označený jako enable (EN) ten slouží jako spínací pro zapnutí, respektive vypnutí modulu. Je-li na něm hodnota high je modul zapnutý. Mezi piny 3 a 4 zapojíme induktor do kterého budeme v podobě magnetického pole ukládat elektrickou energii. Spínané výstupní napětí je na pinu číslo 3. Posledním pinem je pin 5. Ten je zde pro připojení zpětné vazby do obvodu regulátoru. Výstupní hodnotu nastavujeme změnou poměru odporů R1 (Ω) a R2 (Ω). Ty fungují jako dělič napětí. [21]

Vztah pro výpočet výstupního napětí je:

$$\mathrm{V} = \left(1.25 * \left(1 + \frac{R2}{R1}\right)\right) (V) \tag{4.2}$$

### 4.2.7 Arduino MKR GPS Shield




*Obrázek 23: Arduino MKR GPS Shield*

Tento rozšiřující modul je navržený pro desky Arduino MKR a poskytuje jim polohové informace. Je postaven na modulu u-blox SAM-M8Q GNSS (Global Navigation Satellite System). Podporuje tři hlavní polohovací služby GPS, Gloanass a Galileo. Komunikace je možná pomocí sériové komunikace po sběrnici pomocí UART (Universal Asynchronous Receiver-Transmitter) sériového komunikačního protokolu. Jedná se o asynchronní provoz, kde se nepoužívá hodinový signál pro synchronizaci. Pro zjednodušení práce s touto deskou je použita knihovna Arduino_MKRGPS.h. Pomocí této knihovny můžeme vyčítat data z modulu, jako je zeměpisná výška, šířka, nadmořská výška, rychlost a počet právě přijímaných satelitů. Později bude použito v kódu a pomocí dashboard a jejich widgetů bude zobrazena přesná poloha pomocí Google maps. [22]

## 4.3 Programové vybavení a vývojové prostředí

Celý projekt bude vytvořen v prostředí Arduino Cloud na webových stránkách https://app.arduino.cc/. Na těchto stánkách je několik důležitých funkcí, které budou použity při tvorbě projektu. Jeho hlavní funkce jsou:

- **Webové IDE** (Integrated Development Environment)
  Slouží nám k vytváření, kompilaci a nahrávání kódu na vývojovou desku. Jsou zde všechny potřebná knihovny pro komunikaci s různými periferiemi a zařízeními.
- **Správa zařízení**
  Z tohoto místa sledujeme stav jednotlivých zařízení. Nastavujeme zde jejich síťové parametry a provádíme aktualizaci firmwaru.
- **Propojování s Cloudovými službami**

Arduino Cloud umožňuje pomocí API (Application Programming Interface) propojení na různé Cloudové služby.

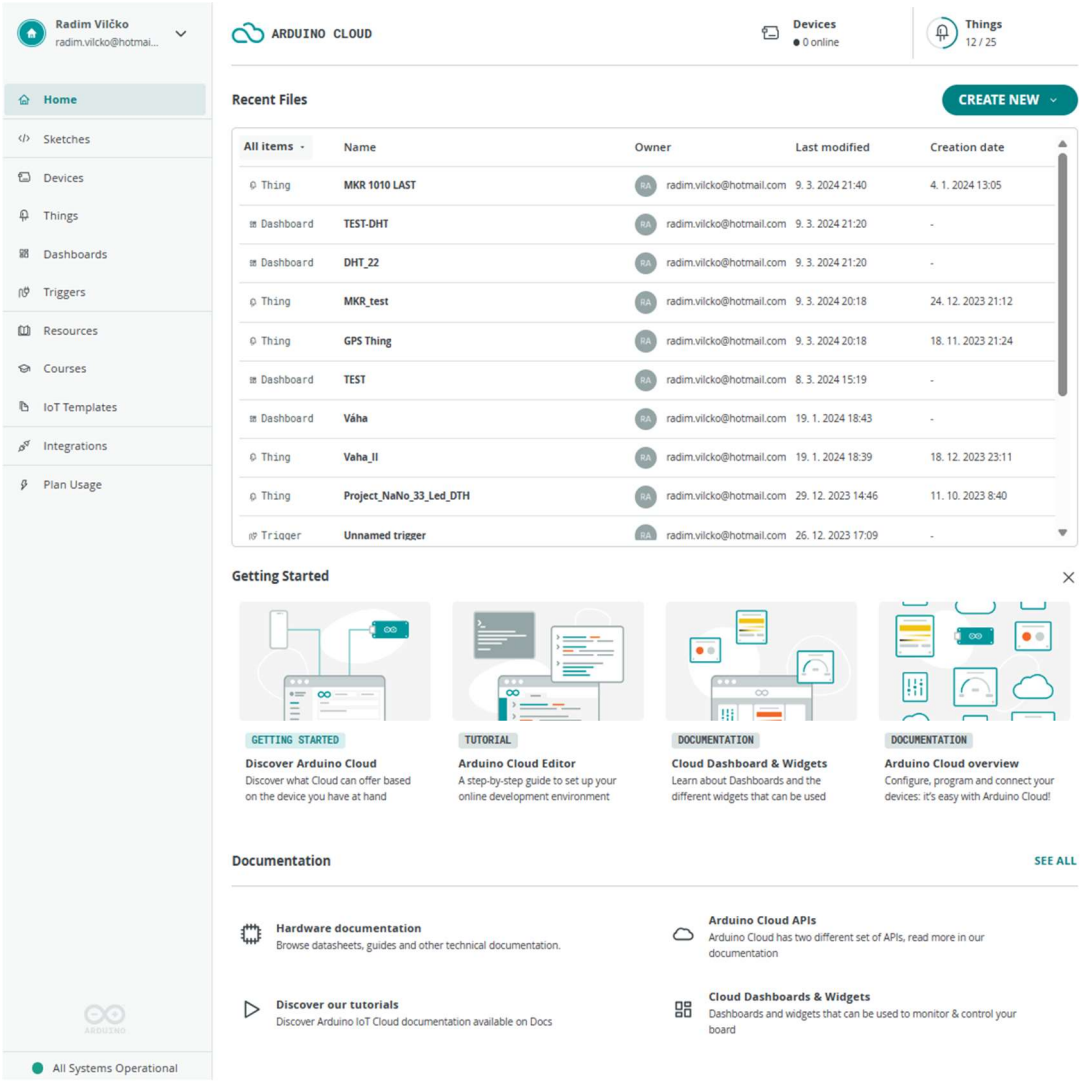
- **Sběr a analýza dat**

  Další vlastností Arduino Cloudu je možnost uchovávat data a interpretovat je pomocí tzv. dashboardů. Později bude několika těchto dashboardů použito v práci.

- **Vzdálené ovládání a automatizace**

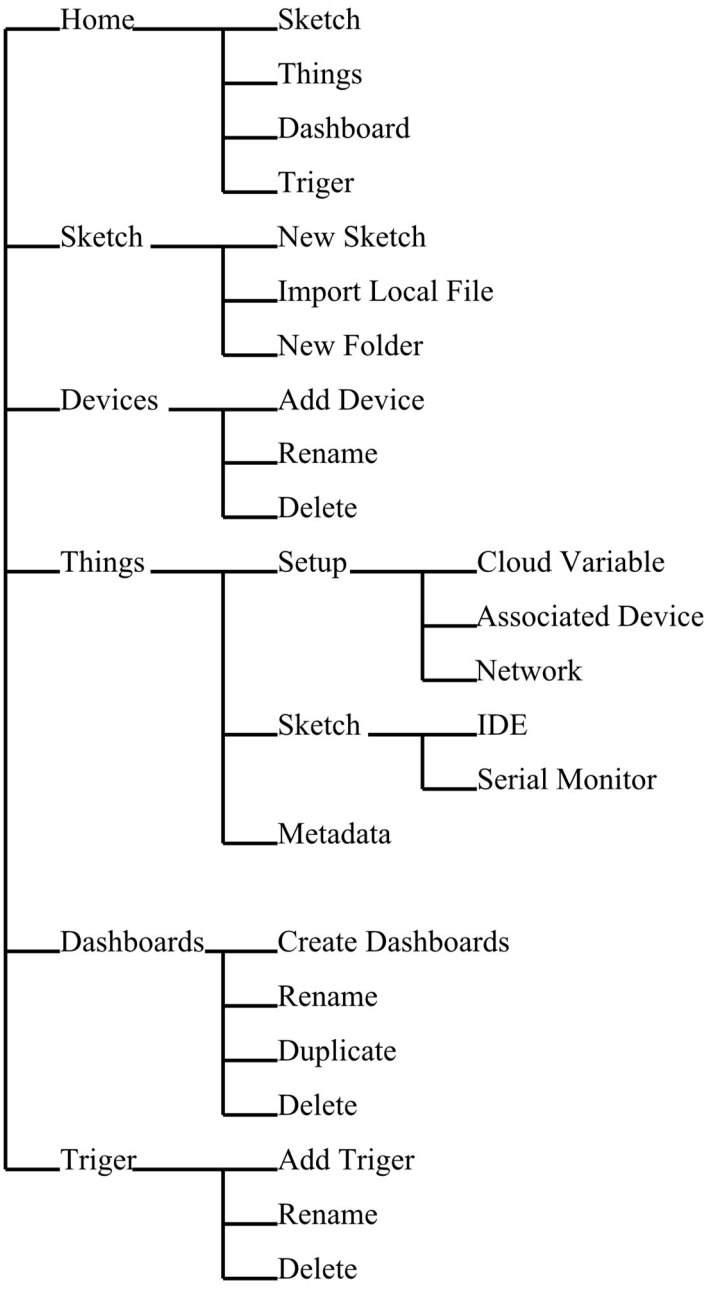
  A v neposlední řadě zde nastavujeme i chování systému v závislosti na vnější podněty. To znamená můžeme automaticky reagovat na nějakou událost anebo pomocí ruční zásahu operátora vzdáleně ovládat připojená zařízení. [23]

Na obrázku číslo 24 je web Arduino Cloud. Zde jsou na levém panelu jednotlivé komponenty pro práci s deskami kompatibilními s tímto webem.




*Obrázek 24: Arduino Cloud*

Další obrázek zobrazuje stromovou strukturu Arduino Cloudu.




*Obrázek 25: Stromová struktura Arduino cloud*

**Home** je výchozí rozhraní, kde vidíme nedávno použité projekty. Umožňuje nám rychlí návrat k nedávno použitým projektům. Jednotlivé položky budou popsány v dále v textu.

**Sketch** je webové IDE, pomocí kterého tvoříme zdrojový kód pro ovládání a monitorování připojené desky. Programovací jazyk vychází ze syntaxe C++. Dále pro zjednodušení vývoje je k dispozici mnoho rozšiřujících knihoven. Ty nám v projektu

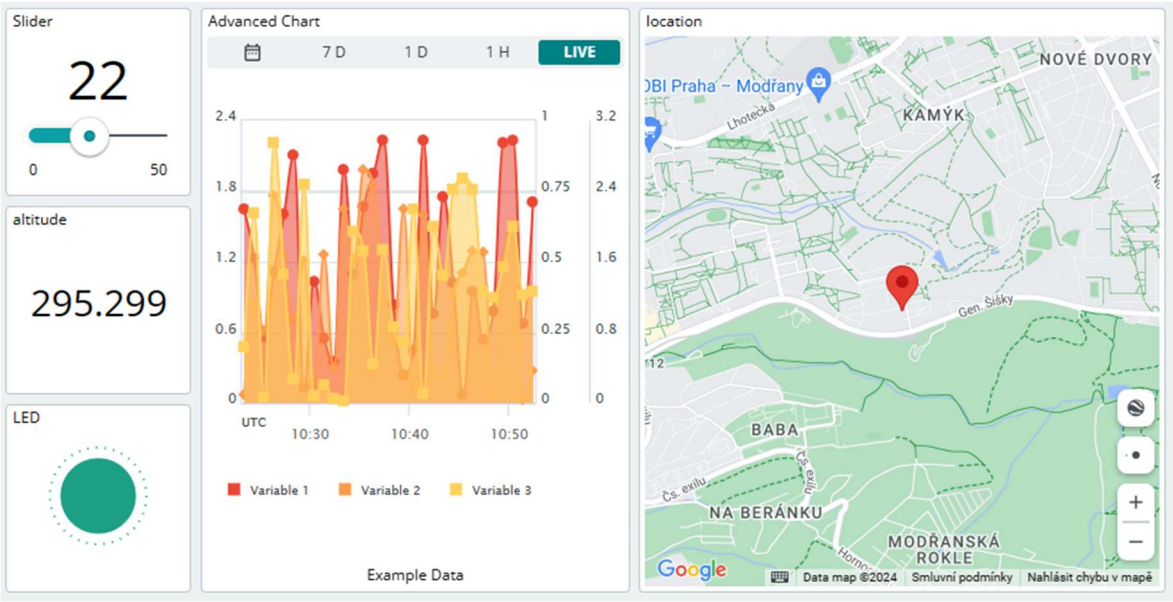
pomohou s velmi jednoduchou implementací jednotlivých periferií. Dále zde provedeme kompilaci programu a jeho nahrání do mikrokontroleru.

**Devices** zde vidíme jednotlivé zařízení, která jsou připojena do cloudu. Je zde také jejich status to, zda jsou online či nikoli a o jakou vývojovou desku se jedná a s jakým projektem je spárována. V pravém horním rohu je tlačítko pro přidávání dalších zařízení.

**Things** jsou jednotlivé projekty. Tato položka je zde pro samotné propojení konkrétní desky Arduino s napsaným kódem. Dále zde vytváříme jednotlivé cloudové proměnné. A další důležitou funkcí této položky je nastavení síťových parametrů. V našem případě se jedná o parametry wifi připojení název wifi a heslo pro připojení.

Pro komunikaci s připojenými Arduino deskami je nutné, aby na počítači běžel Arduino Create Agent. Pomocí něj nahráváme zkompilovaný kód do desek. Tento agent je k dispozici na adrese https://create.arduino.cc/getting-started/plugin/download.

**Dashboards**, zde se vytváří webový interface pro monitorování a ovládání IoT zařízení. Propojujeme jednotlivé cloudové proměnné, které navážeme na objekty takzvané widgets. Ty jsou například typu graf, hodnota, přepínač, mapa a další viz obrázek 26.




*Obrázek 26 Arduino cloud widgets on dashboard ukázka*

**Triggers** je poslední položka kterou budeme používat. Jedná se o funkcionalitu, která nám pomůže se zasíláním notifikací. Můžeme zde vygenerovat notifikace zasílání emailu nebo notifikace zasílané do aplikace v mobilním telefonu. Typické použití je například jestliže teplota stoupne, respektive klesne nad námi stanovenou mez zašli o tom email nebo pošli notifikaci do mobilního zařízení.

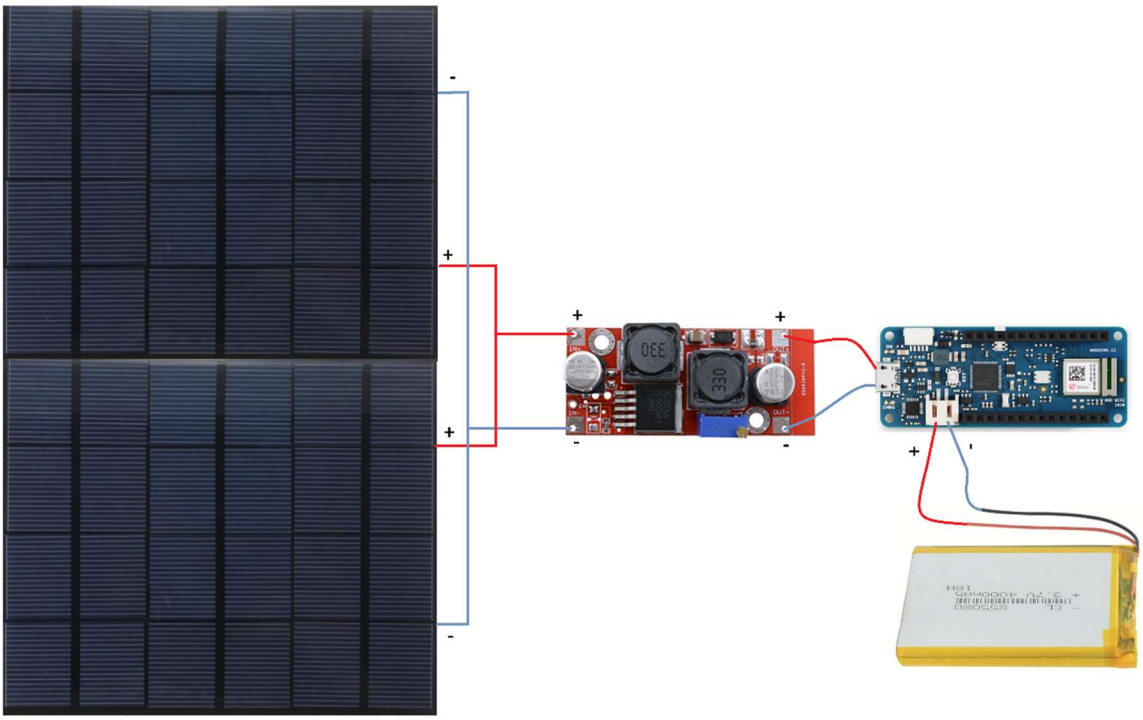
# 5 Vlastní práce

Toto téma bude rozebráno v několika oblastech. Jednak samotné hardwarové sestrojení a schéma zapojení jednotlivých komponent. Tak potom zaměření se na samotný zdrojový kód a propojení s Arduino cloudem.

Popíšeme napájecí řetězec, který se skládá z dobíjecí soustavy fotovoltaického panelu, měniče napětí a Li-Pol akumulátoru. Dále to budou jednotlivé senzory hmotnosti, teploty, vlhkosti a jako poslední GPS Shield. To vše bude propojeno pomocí kabelů s konektory a umístěno ve vhodných pouzdrech s patřičným IP krytím proti vlhkosti a prachu. Vše bude připojeno do Arduina MKR 1010 Wifi to následně bude zaintegrováno do služby v Arduino cloudu.

## 5.1 Napájení

Napájení bylo vyřešeno z důvodů ostrovního fungování Li-Pol akumulátorem o celkové kapacitě 4000mAh a jmenovitém napětí 3,7V. Jelikož deska Arduino MKR 1010 je vybavena konektorem pro připojení akumulátoru typu JST S2B-PH-SM4-TB (samice) byla i baterie vybavena tímto typem konektoru JST-PHR (samec) s roztečí 2 mm viz obrázek 28. Maximální povolený proud na tomto typu konektoru je 2 A a 100 V AC/DC. [24]




*Obrázek 27: Schéma zapojení napájení*

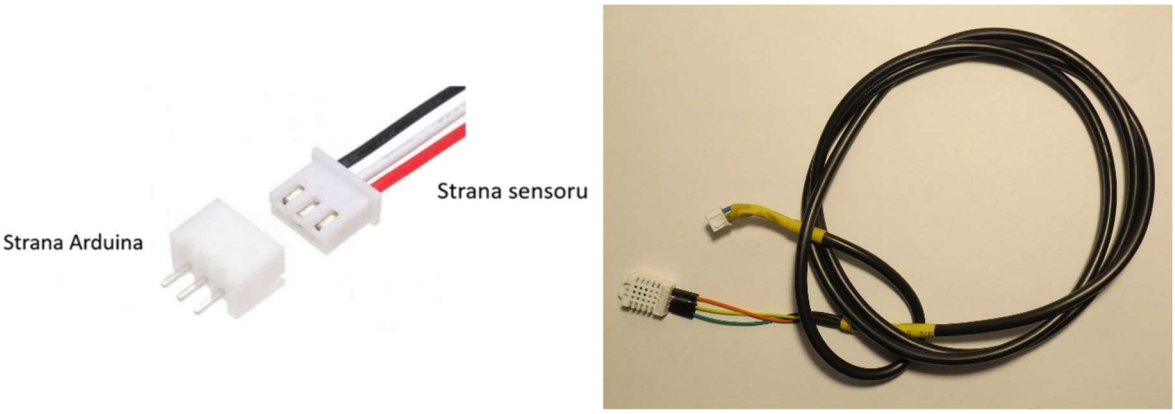
*Obrázek 28: Konektor JST PHR 2 mm a JST S2B-PH-SM4-TB*

V první fázi testování byl použit pouze jeden panel, ale to se ukázalo jako nedostatečné. Za ideálních podmínek stačila soustava napájet desku, ale nebyl již dostatečný výkon k plnému dobití baterie. Místo koupě nového výkonnějšího solárního panelu jsem přistoupil k rozšíření o druhý stejný. Bylo provedeno paralelní zapojení těchto panelů. S toho vyplívá že při zvážení všech parametrů se výstupní napětí nezměnilo a zůstává 6 V na jejich výstupu. Maximální proud z jednoho panelu je dle dokumentace 550 mA. Na dva panely při paralelním zapojení je maximální proud 1100 mA. Jelikož boost-buck step up/down DC-DC XL 6009 modul dokáže pracovat na vstupu až z 3 A je zachována dostatečná proudová rezerva, a to až skoro troj násobná.

V dalším kroku bylo na napěťovém měniči nastaveno výstupní napětí do zátěže na 5 V. To znamená že byla nejprve nastavena nejnižší možná výstupní hodnota napětí, aby nedošlo ke zničení mikrokontroleru. Samozřejmě toto nastavení probíhalo za slunného dne, kdy byl předpoklad že solární panely budou generovat maximální množství elektrické energie. Poté byl měnič připojen na vstup USB Arduina a trimerem pomocí šroubováku byla nalezena optimální hodnota 5 V na vstupu mikrokontroleru. Tato konfigurace již byla dostatečná a zařízení fungovalo bez problémů.

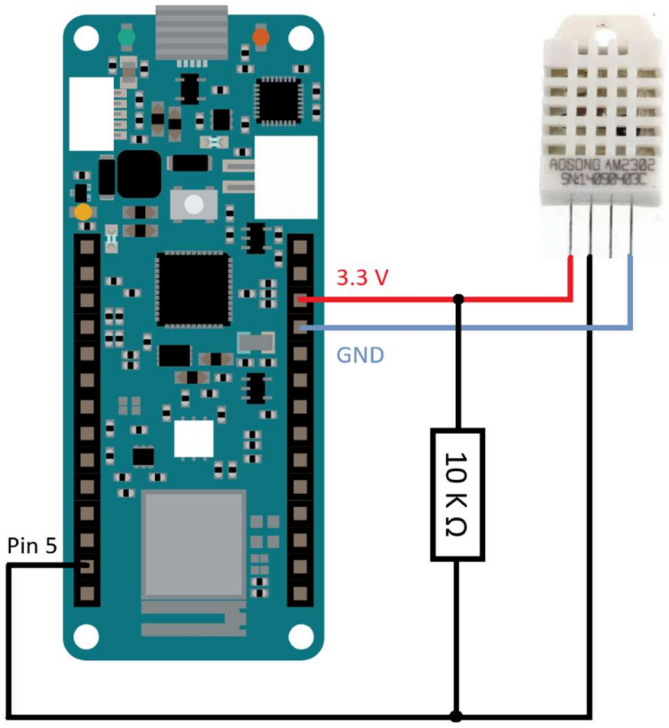
## 5.2 Zapojení DHT22 senzorů

Jak již bylo zmíněno víše, pro měření hodnot teploty a vlhkosti byl zvolen sensor DHT_22. Ten je pro naše potřeby více než dostatečný. Jednak je to jednoduchá součástka, se kterou se výborně pracuje má vynikající podporu v projektech Arduina v podobě knihoven. A zároveň je velmi levná a v podstatě velice odolná. Pro jeho spojení s mikrokontrolerem Arduino byly zvoleny konektory JST 3 pinové s roztečí 2,54 mm. Na straně kabelu byl samec na straně Arduina samice to reprezentuje obrázek číslo 29.




*Obrázek 29: Konektory JST 3 pin 2,54 mm s kabelem*

Na následujícím obrázku je schéma zapojení. Pro ilustraci je zde pouze jeden senzor v projektu jich bude použito hned několik. Další budou sloužit například pro monitorování vnější teploty a jiné pro kontrolu teploty uvnitř úlu. Jak je patrné z obrázku je mezi datovým a napájecím pinem připojen ještě 10 kΩ rezistor. Jedná se takzvaný pull-up rezistor. Ten pomáhá zajistit spolehlivou komunikaci mezi sensorem a mikrokontrolerem. Eliminuje vliv šumu na jeho vstupní pin a tím chybné interpretaci stavu.




*Obrázek 30: Schéma zapojení DHT 22*

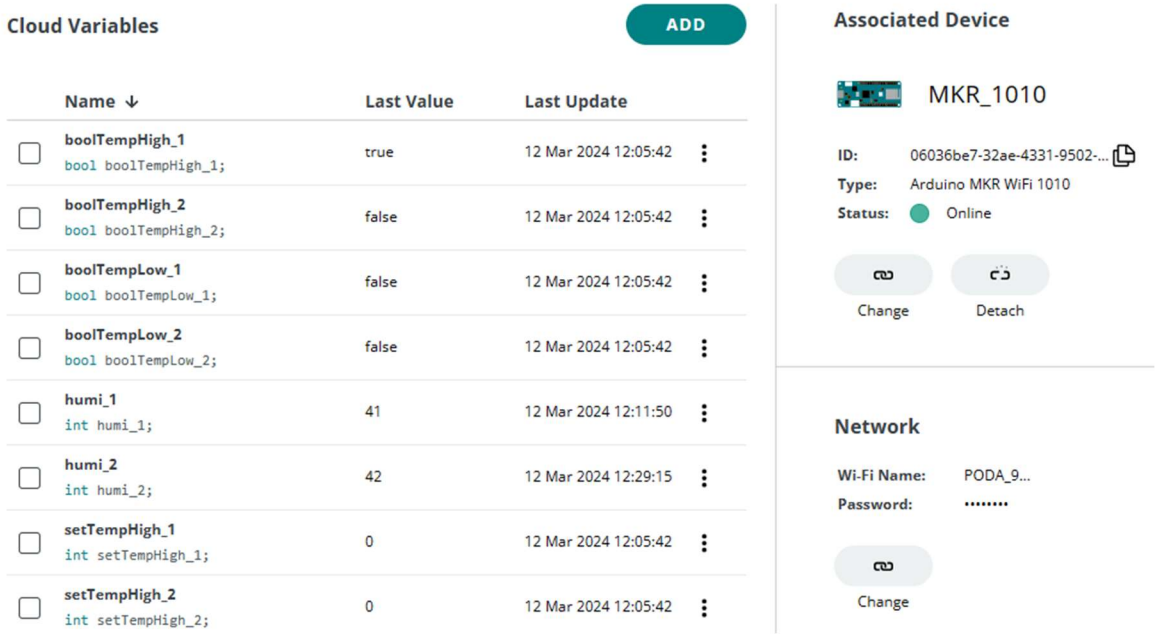
## 5.3 Program pro DHT22

Jakmile máme hotové hardwarové řešení budeme pokračovat v psaní kódu. Pro psaní se velice osvědčilo webové IDE. Nebylo třeba hlídat aktualizace potřebných knihoven a vždy bylo po ruce vše potřebné. Webové IDE vše podstatné vyřešila za nás. Jednalo se zejména o rutiny, jak připojit mikrokontroler pomocí wifi k síti, definovat cloudové proměnné se kterými budeme dále pracovat a připojit požadovanou desku.

V prvním kroku se vytvořil v menu Things nová instance projektu. V tomto případě byl její název DHT_22_Temp_Humi. Nejedná se o nic podstatného jen nám tento název usnadňuje orientaci v různých projektech. Jako další byl zadán set cloudových proměnných požitých v mém kódu.

Je potřeba v této sekci asociovat rovněž zařízení, které chceme s tímto projektem propojit. Mnou zvolené Arduino připojíme pomocí mikro USB kabelu s počítačem. Na tomto počítači musí být spuštěn Create Arduino Agent. Ten je nutný pro komunikaci mezi počítačem a Arduino deskou po sériové lince.

Poslední, co zde provedeme je nastavení parametrů pro komunikaci s wifi routerem, který nám zprostředkovává konektivitu směrem do internetu. Nastavíme zde Wi-Fi jméno a heslo pro připojení. Na následujícím obrázku je část proměnných s asociovaným zařízením a nastavením pro Wi-Fi.




*Obrázek 31: Založení projektu v Arduino Cloudu*

### 5.3.1 Vývojové diagramy pro senzor DHT22

Na začátku každého programu, který bude do mikrokontroleru nahrán, vložíme název souboru thingProperties.h. Ten nám webové IDE vygeneruje samo. Nachází se v něm potřebné knihovny pro snadnou integraci do internetu věcí. Bude volán pomocí metody později v kódu. Jedná se o knihovny *ArduinoIoTCloud.h* a *Arduino_ConnectionHandler.h*. První ze zmíněných knihoven slouží pro práci s definovanými proměnnými, které můžeme aktualizovat nebo sledovat jejich stav. A tak reagovat na změnu v reálném čase. Druhá poskytuje abstrakci pro různé typy síťového připojení. Uživatel je odstíněn o různých typů připojení a nastavení je tak mnohem jednodušší. Jedná se například o síťové protokoly jako je ethernet, WiFi, GSM a podobně. V mém případě je zde definováno síťové rozhraní pomocí WiFi. Proto je v tomto souboru rovněž uloženo jeho nastavení. Jedná se pouze o dvě položky. SSID název WiFi sítě a její heslo. Vše ostatní za nás řeší *Arduino_ConnectionHandler.h* jak bylo zmíněno již víše.

Pro navázání konektivity pomocí WiFi zavoláme funkci *WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS)* s parametry.

Dále jsou zde takzvané callbacky. Jedná se o funkce, jež jsou volány pokaždé když se změní hodnota definovaných proměnných. V tomto případě nastavujeme hodnotu horní a dolní meze pro sledování kritických hodnot pomocí webového rozhraní Arduina. Tyto hodnoty budou dále použity v programu pro zasílání notifikací.

Funkce *initProperties(),* která dále následuje inicializuje proměnné a jejich vlastnosti pro synchronizaci s Arduino IoT Cloud. Nastavuje, zda jsou proměnné pouze pro čtení (READ), nebo umožňují čtení a zápis (READ/WRITE), a určuje, jaké funkce mají být volány při změně hodnoty proměnných. V tomto případě jsou zde definovány proměnné pro ukládání naměřených hodnot teploty a vlhkosti z jednotlivých sensorů. Zde jsou dva příklady. První *ArduinoCloud.addProperty(temp_1, READ, ON_CHANGE, NULL)* ten vyčítá hodnotu teploty z prvního sensoru a uloží ji do proměnné. Druhý *ArduinoCloud.addProperty(setTempHigh_1, READWRITE, ON_CHANGE, onSetTempHigh1Change)* přečte hodnotu z webu a uloží ji do proměnné.

Vše výše popsané souvisí s blokem inicializace ThingProperties.h jež je součástí diagramu číslo 1.

Po této inicializaci načteme knihovnu DHT.h. Ta nám zprostředkuje komunikaci na sensor DHT22 (AM2302). Pro její potřeby musíme dále nadefinovat piny, na které bude

sensor fyziky připojen a typ sensoru. To je z důvodu že tato knihovna je společná i pro starší typ sensoru DHT11. Nakonec vytvoříme instance sensorů a předáme definice.

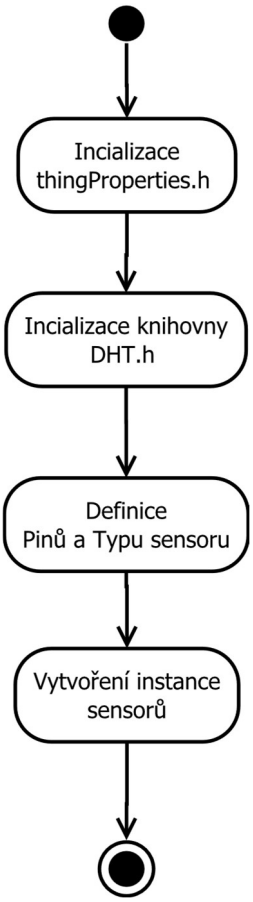



*Diagram 1: Inicializace sensoru DHT*

Program dále pokračuje zaváděcí funkcí *setup()*. Ta je volána pouze jednou, a to na začátku programu. V jejím těle navazujeme sériovou komunikaci s Arduinem pomocí smyčky čekáme, dokud není úspěšně navázána. Dále voláme funkci *initProperties()*. Ta je definována, jak jsme psaly víše v souboru *thingProperties.h* a nastaví nám potřebné proměnné a funkce pro práci s cloudem.

Inicializujeme spojení mezi Arduino zařízením a Arduino IoT Cloud přes definované síťové připojení pomocí funkce *ArduinoCloud.begin(ArduinoIoTPreferredConnection)*. *ArduinoIoTPreferredConnection* je instance třídy pro správu připojení, typicky nastavená pro WiFi nebo jiný typ síťového připojení, s detaily jako jsou SSID a heslo. V tomto kroku se tedy nejprve inicializuje spojení pomocí přes WiFi a po té provede přes něj propojení Arduina s cloudem.

V případě že je vše v pořádku vypíše se informace o navázání spojení na konzoli a od této chvíle máme trvalé spojení do cloudu. V opačném případě se rovněž vypíšou informace a program se ukončí.

Pro výpis debuggováních informací slouží funkce setDebugMessageLevel(2) kde číslo znamená úroveň. Ty jsou 0 až 4, kdy čtvrtá úroveň je nejpodrobnější.

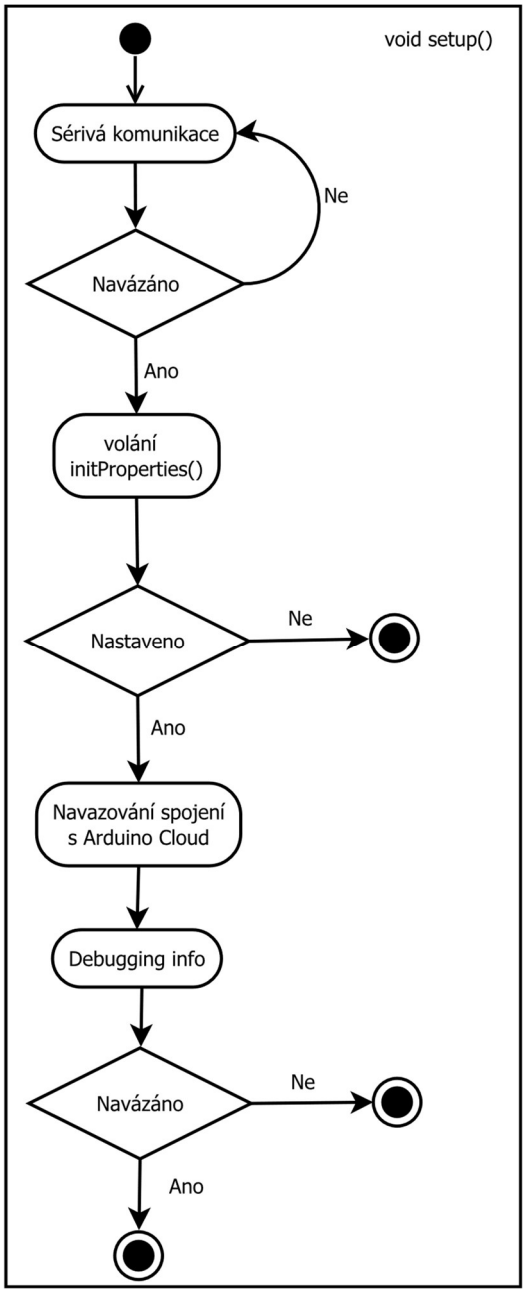



*Diagram 2: Funkce setup()*

Funkce `Loop()` je vykonávána nepřetržitě cyklicky. Z této smyčky jsou v tomto případě neustále volány tři metody. První volání funkce `myDHT()` je zde pouze pro vyčítání hodnot ze sensoru a ukládání do proměnných. Další dvě *myTrigerTempHigh()* a *myTrigerTempLow()* jsou pomocné funkce, které budou použity pro nastavení thresholdů. V tomto případě se na obou sensorech bude tetovat, zdali nebyla překročena mezní hodnota teploty, kterou chceme hlídat. A to jak směrem na horu, tak i dolů. V případě že ano zašle o tom IoT Cloud notifikaci emailem nebo mobilní aplikace.

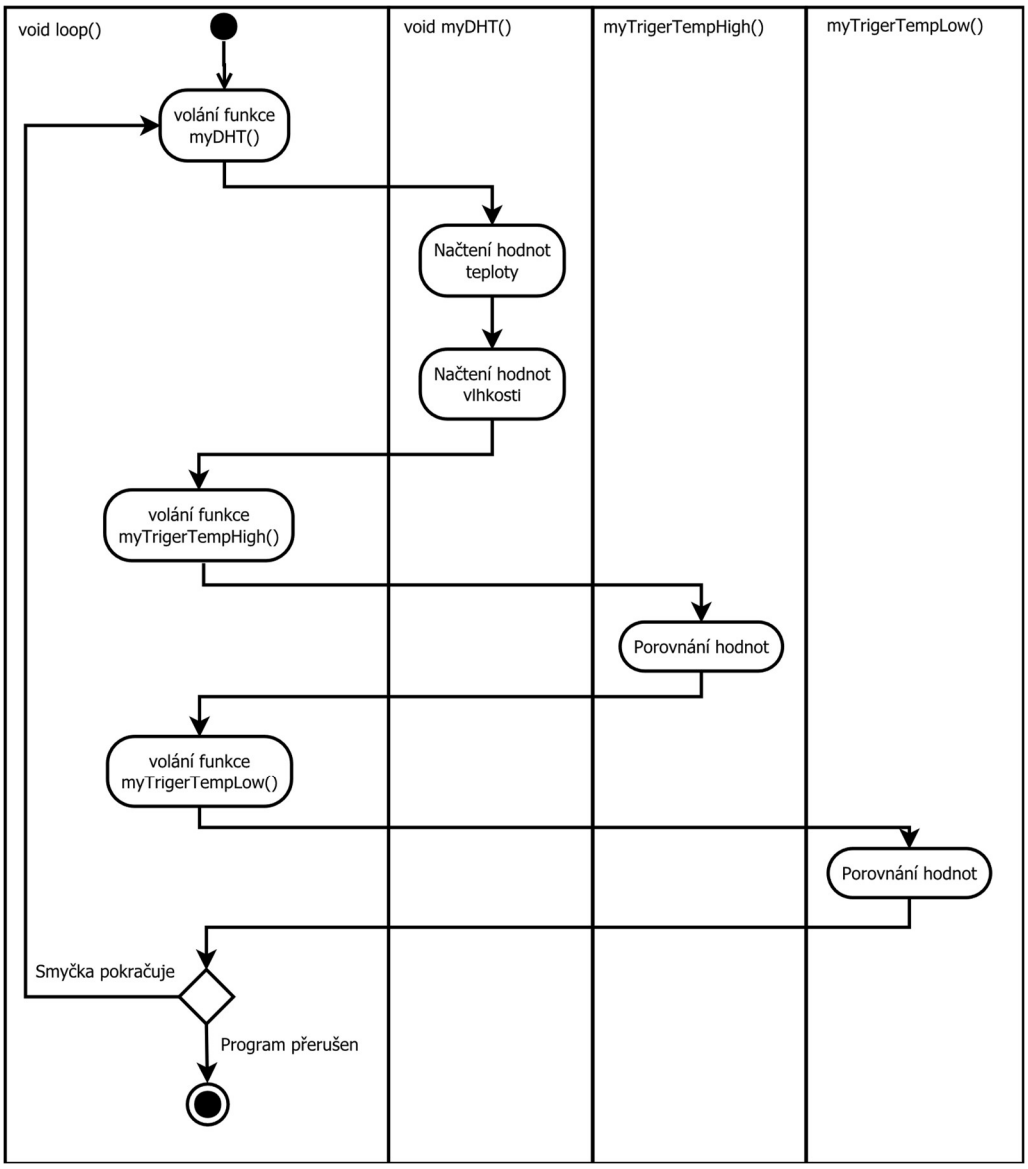



*Diagram 3: Funkce loop*

Z důvodů zasílání notifikací byla v programu navrhnuta funkce *myTrigerTempHigh()*. Ta je zde proto aby monitorovala měřené hodnoty s hodnotami, které nastavil uživatel. Tato funkce je velice jednoduchá v pravidelných intervalech je volána a provádí porovnání naměřených a referenčních hodnot. Jak vyplívá z následujícího diagramu. Pokud je hodnota sensoru vyšší, než reference nastaví se booleovská hodnota na true. Ta je později zpracována v Arduino cloudu a slouží jako příznak pro vygenerování notifikace. Ta může být odeslána emailem nebo zaslána do aplikace v mobilním telefonu. Později bude návrh ukázán podrobněji.

Jelikož nechceme monitorovat pouze překročení maximální hodnoty, je stejným způsobem naprogramována i funkce *myTrigerTempLow()*. Její diagram by byl naprosto totožný s malým rozdílem. Pouze se prohodily true a false mezi sebou. Tím bylo zajištěno, že při poklesu teploty dostaneme příznak pro notifikaci na kriticky nízkou hodnotu.

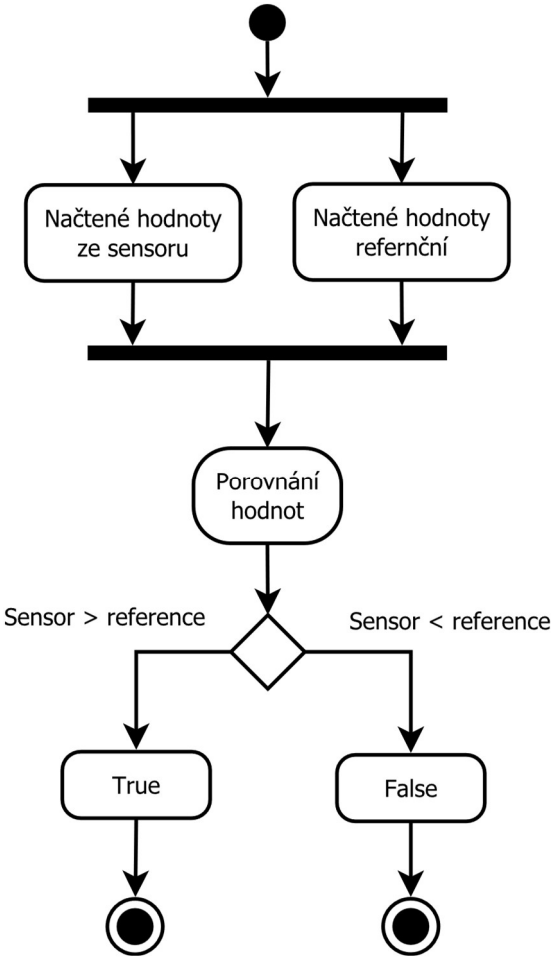
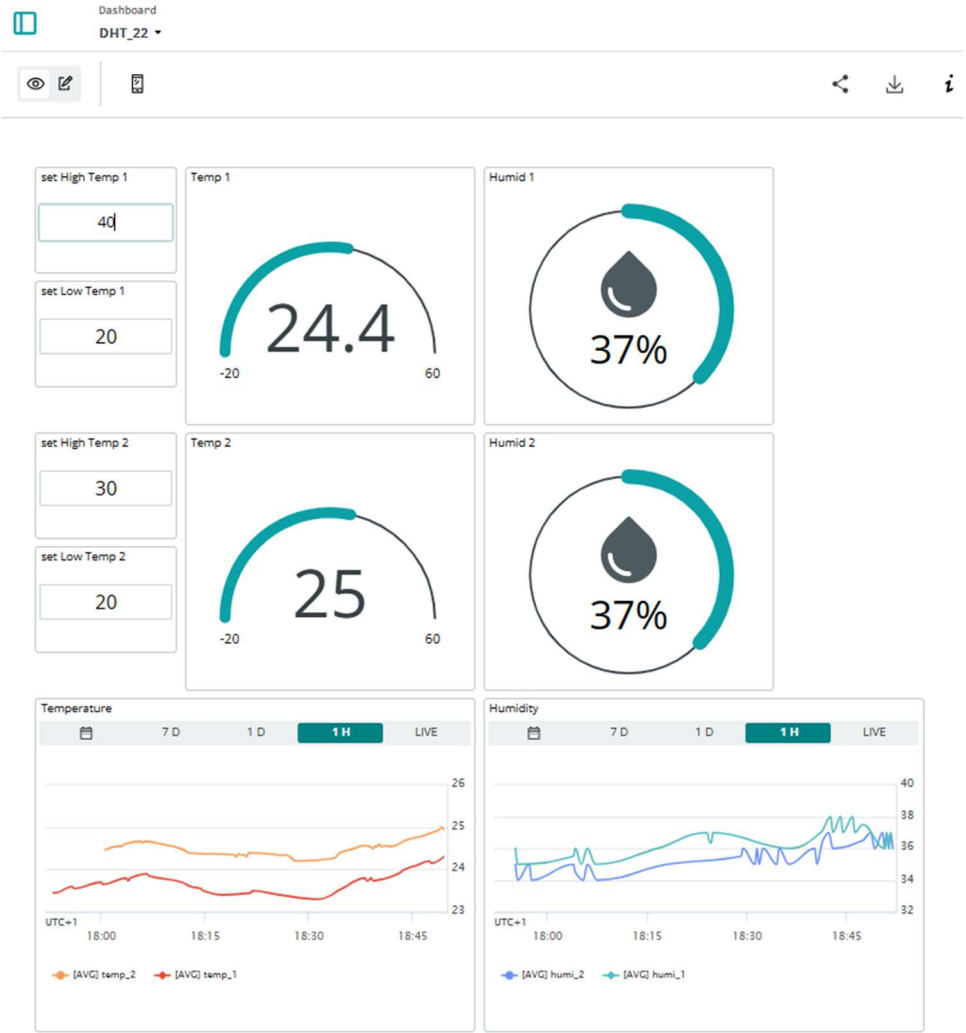



*Diagram 4: Funkce trigger*

### 5.3.2 Prezentační vrstva webové rozhraní DHT22

Jako poslední zde bude ukázka webového rozhraní. Na něm je možno námi naměřené hodnoty zobrazit uživateli. Vytvoření dashboardu v IoT Claudu Arduina je velice jednoduché. Na hlavní stránce v levém menu vybereme položku Dashboards. Pokud nemáme vytvoříme nový. Zadáme jméno v mém případě DHT_22. Otevře se nám nová webová plocha. V levém horním rohu máme přepínač pro náhled a editační mód. Přepneme do editace. Položky se rozšíří o tlačítko ADD. To nám umožňuje přidávat celou škálu jednotlivých widgetů. Jedná se například o zobrazovací prvky hodnot, přepínače pro vzdálené ovládání, různé druhy grafů, mapy a mnoho dalších.

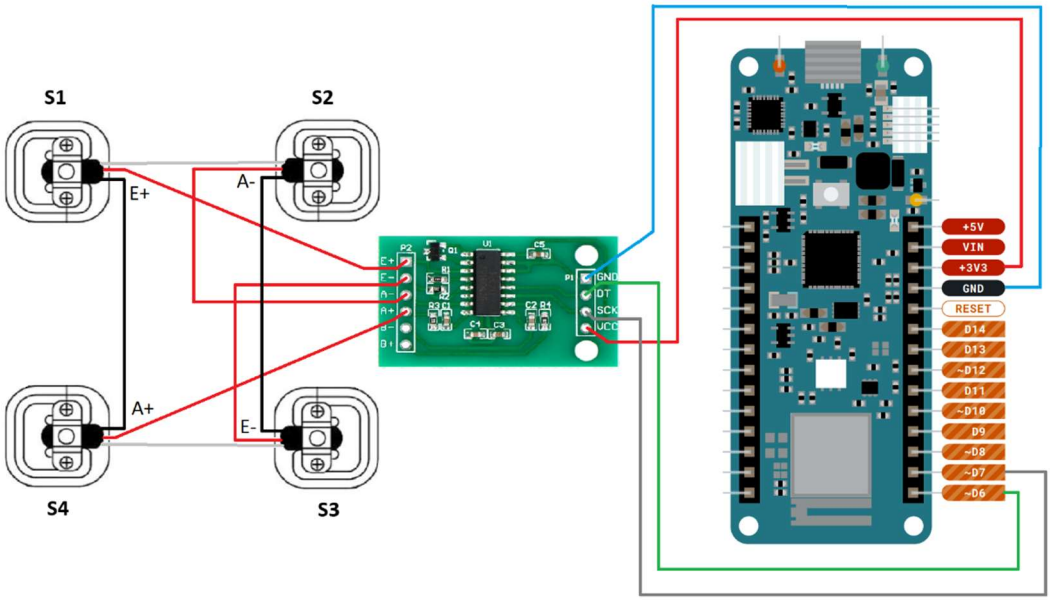
V tomto editačním módu umístíme požadované prvky přehledně na plochu. Ukázka mého rozvržení je na následujícím obrázku. Poté je nutné jednotlivé prvky navázat na proměnné v našem programu. To samé provedeme i v náhledu pro mobilní zařízení.




*Obrázek 32: Webové rozhraní teploty*

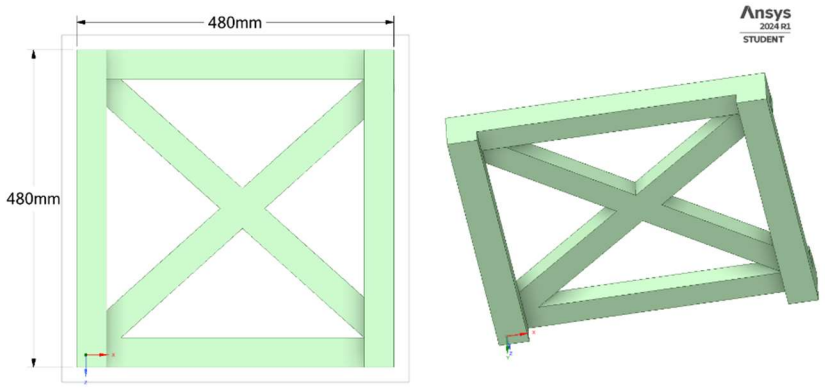
## 5.4 Tenzometrická váha

Pro měření hmotnosti úlu byly zvoleny tenzometrické sensory. Jim byla věnována celá kapitola 4.2.1. Zde pouze pro ilustraci je vložen obrázek, jakým způsobem je celá měřící soustava fyzicky zapojena.




*Obrázek 33: Fyzické zapojení váhy*

Kabeláž byla vyrobena opět na míru. Použit byl čtyř žilový vodič. Směrem k mikrokontroleru zakončen konektorem JST samec 4 pinový s roztečí 2,54 mm. Druhý konec směrem k AD převodníku byl napájen přímo na jeho konektory. Z převodníku směrem k sensorům vše bylo napájeno na pevno. Pro tenzometry byl vyroben podstavec z dřevěných profilů o velikosti 25 mm x 45 mm. Další obrázek jej zobrazuje, vytvořen byl pomocí Ansys Space Claim 2024.




*Obrázek 34: Podstavec pod úl*

### 5.4.1 Diagram váhy

Tato ukázka začíná stejně jako předchozí program navržený pro sensory teploty a vlhkosti DHT 22. Provede se inicializace souboru ThingProperties.h. Opět se pomocí tohoto zaváděcího souboru deklarují proměnné. Dále naváže konektivita prostřednictvím WiFi a definují funkce pro příjem a odesílání dat směrem do claudu.

Další krok spočívá v inicializaci knihovny HX711.h. Ta nám umožní pohodlně komunikovat s AD převodníkem a pomocí sériové komunikace. Pak jsou definovány příslušné piny na desce, a nakonec založeno několik pomocných proměnných například pro kalibraci nebo hodnotu odečtenou s AD převodníku.

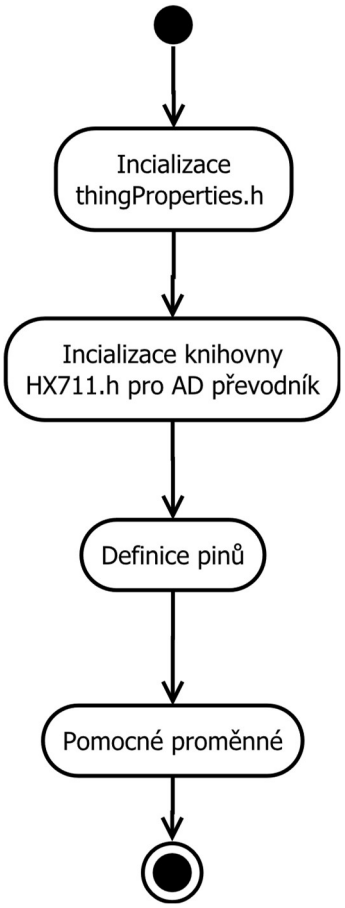



*Diagram 5: Inicializace a načtení knihovny HX711.h*

První zaváděcí funkce *setup()* je totožná s předešlou ukázkou. V této části je pouze jeden rozdíl a to v řádku *scale.begin(dataPin, clockPin),* Kdy pomocí této funkce inicializujeme komunikaci s převodníkem. Tato procedura následuje okamžitě za initProperties(). Jak je patrné předáme této funkci pouze dva parametry definované v předchozí části. První je datový pin a druhý hodinový nutný pro správné fungování sériového přenosu dat.

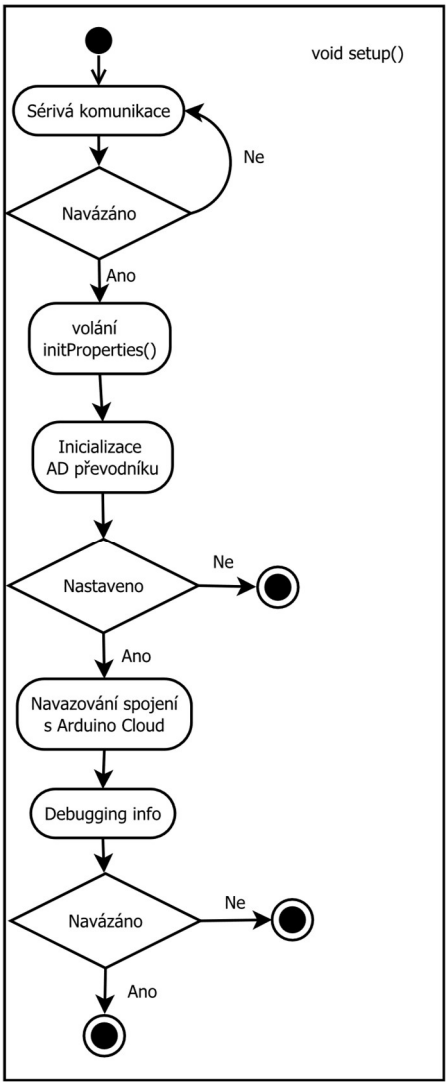



*Diagram 6: Funkce setup() pro váhu*

Funkce *loop()* v tomto případě provádí opakované volání funkce pro obnovu dat v IoT cloudu a volání funkce *myScale*(). Funkce *myScale()* pouze vyčítá hodnoty z převodníku a převádí je na jednotky hmotnosti. Pro přesné měřených veličin je nutné váhu před použitím jinak vynulovat. K tomu nám slouží takzvaná tárovací funkce a správně zkalibrovat. K tomu budou k dispozici další dvě pomocné funkce, které nám toto pomohou nastavit.

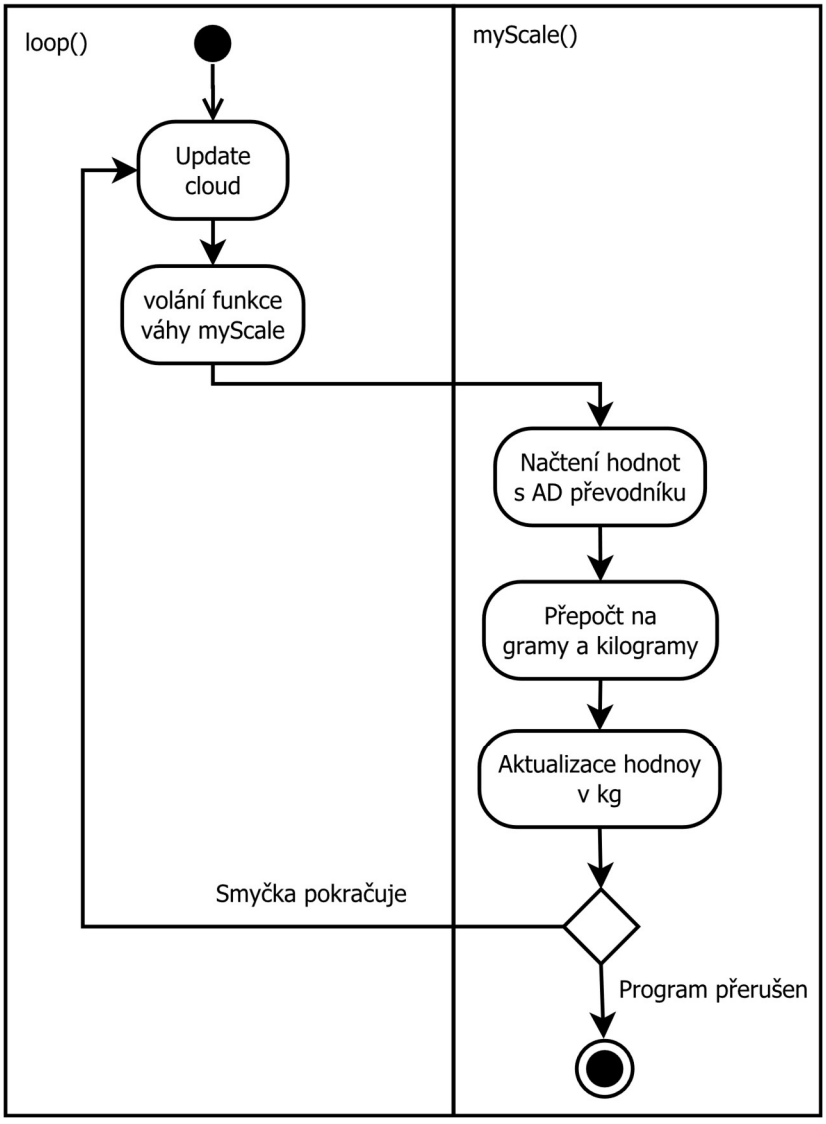



*Diagram 7: Funkce loop() pro váhu*

Další důležitou částí programu je kalibrační funkce. Ta je spouštěna klikem na kalibrační tlačítko, které je součástí webového rozhraní. Pro kontrolu se rozsvítí led dioda tamtéž jako kontrola správně proběhlé kalibrace. Hodnota je uložena v proměnné `gramsUnit`. Vznikne poměrem naměřené hodnoty z AD převodníku ku velikosti kalibračního závaží umístěného na váze v gramech. Ta je uložena do proměnné a použita při vážení úlu. Její pomocí převedeme naměřené jednotky z AD převodníku na jednotky hmotnosti.

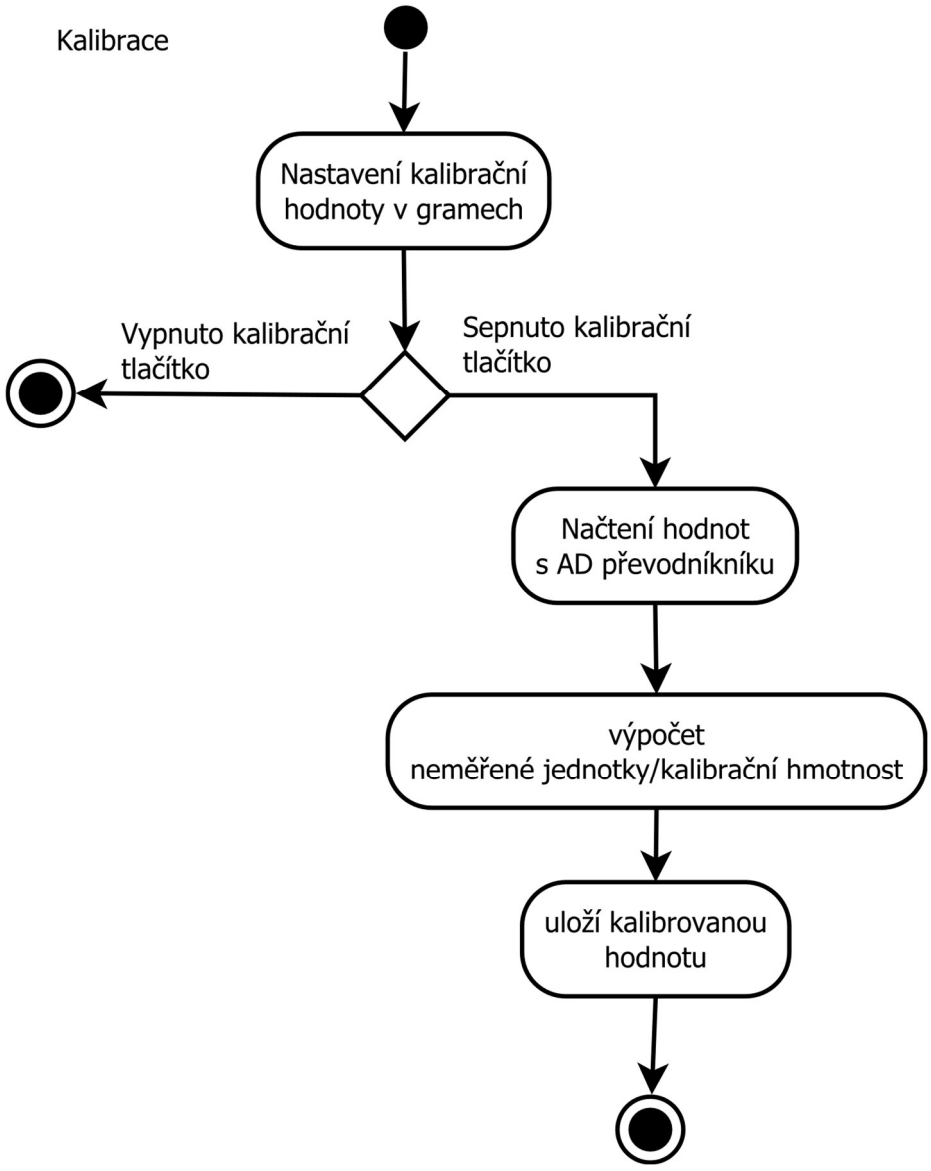



*Diagram 8: Kalibrace váhy*

Pro potřeby přesného vážení je zde rovněž takzvaná tárovací funkce. Ta na váze nastavuje nulovou hodnotu pro kilogramy. A to z důvodů zatížení sensorů již před samotným vážením. Nechceme do měřených hodnot hmotnosti například započítávat hmotnost

samotné váhy a úlu. Proto nejprve váhu zatížíme a provedeme tárování. Hodnota načtená z AD převodníku se přepočítá tak že se jeho výstup vynuluje. Tím máme nastaveno výchozích 0 kilogramů.

Funkce bude vypadat podobně jako předchozí pouze se zde zavolá metoda tare na objektu scale, která je instancí třídy HX711 a provede tárování. To znamená odečte aktuálně váženou hmotnost a nastaví ji na 0. Kontrolní mechanismus je zde stejný jako v předchozím případě pouze se rozsvítí druhá dioda, která je pro tuto funkci určena.

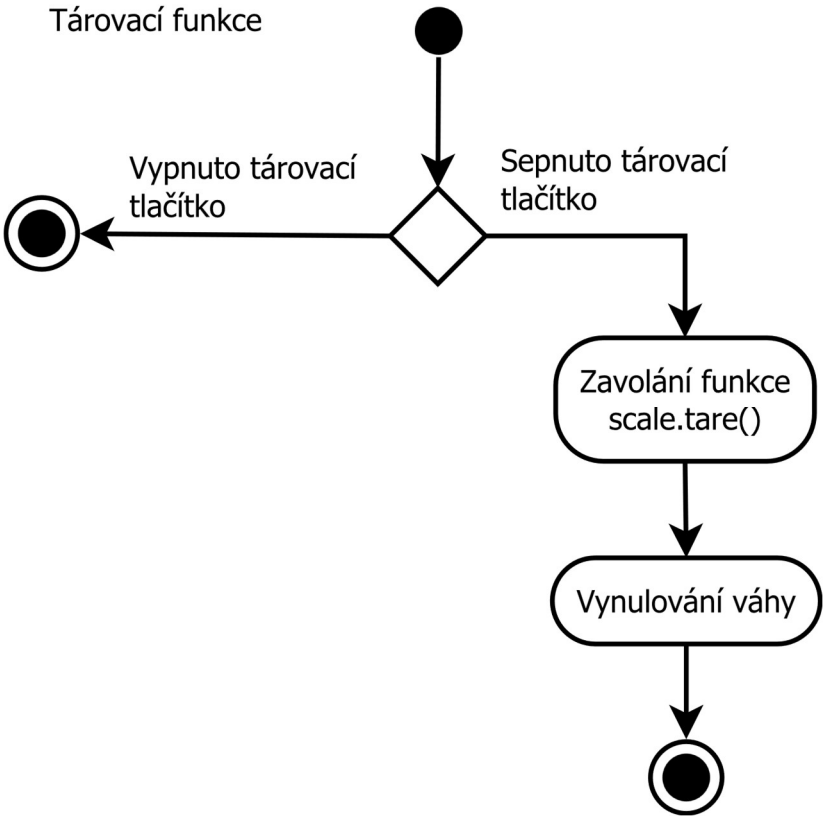
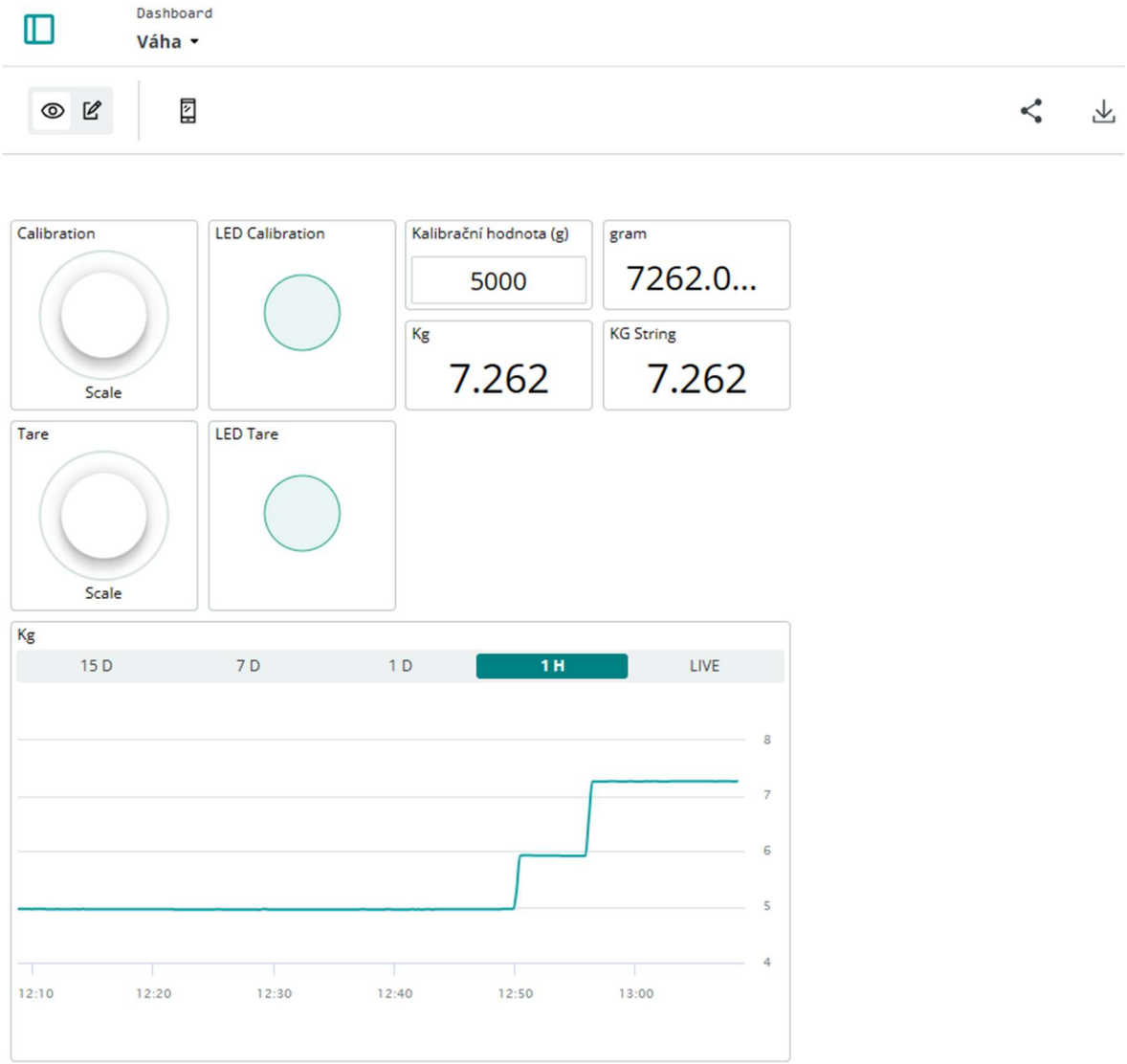



*Diagram 9: Tárovací funkce váhy*

V mém případě používám 5 kilové závaží kterým tuto váhu kalibruji. Jinými slovy celý proces funguje následovně v prvním kroku postavím včelín na konstrukci váhy. Zapnu tárování pomocí tlačítka na webu počkám až se váha vynuluje. Poté vložím na váhu spolu s úlem i toto závaží a provedu kalibraci rovněž z webu. Na webu nastavím kalibrační hodnotu na 5000. Jedná se o hodnotu závaží v gramech. Po signalizaci rozsvícením diody je váha nekalibrována a připravena k vážení. Sejmu kontrolní závaží a v tuto chvíli by měla váha ukazovat hodnoty blízké 0 v kilogramech. Pomocí webového rozhraní je možné nadefinovat jakoukoli hodnotu bude-li k dispozici jiné závaží pro kalibraci.

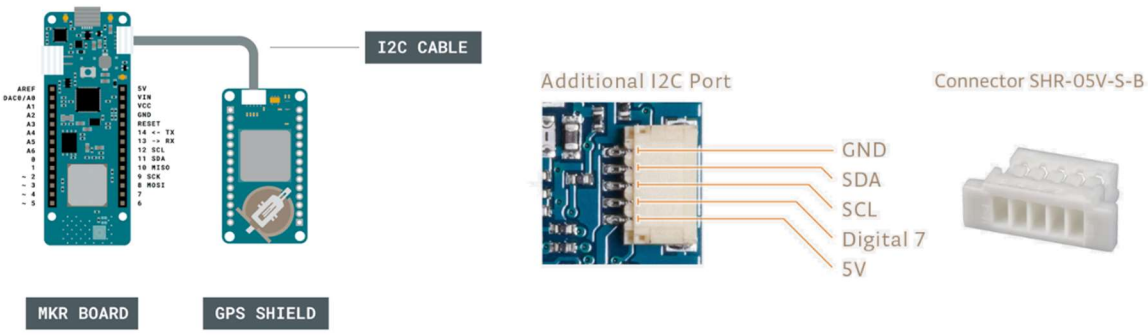
### 5.4.2 Prezentační vrstva webové rozhraní váhy

Pro účely prezentace bylo navrženo webové rozhraní, které je na obrázku číslo 36. Opět maximálně jednoduché a přehledné prostředí. Vše potřebné je na jednom místě. Je zde ovládací tlačítko pro tárování plus jeho kontrolní dioda. Dále kalibrace a její kontrolní prvek. Na spodní straně se nalézá graf, který můžeme přepínat v několika režimech a nastavit tak rozsah viděných dat. Poslední jsou různé zobrazovací prvky aktuálně naměřené hodnoty jak v kilogramech, tak i gramech. V pravém horním rohu je ikona pro stažení uložených dat z cloudu. Tento web je optimalizován i pro malá mobilní zařízení.




*Obrázek 35: Dashboard váha*

## 5.5 MKR GPS Shield implementace




*Obrázek 36: Propojení GPS a Arduina, konektor pro I2C*

Tento modul je originální produkt od firmy Arduino. Proto jeho podpora byla na velmi vysoké úrovni, jak co se týče zpracování, tak podpory v podobě knihovny pro použití a dokumentace na internetu. K tomuto modulu byl dodán kabel s konektory pro jeho propojení s jakoukoli kompatibilní deskou. Na následujícím obrázku je znázorněno její fyzické spojení.

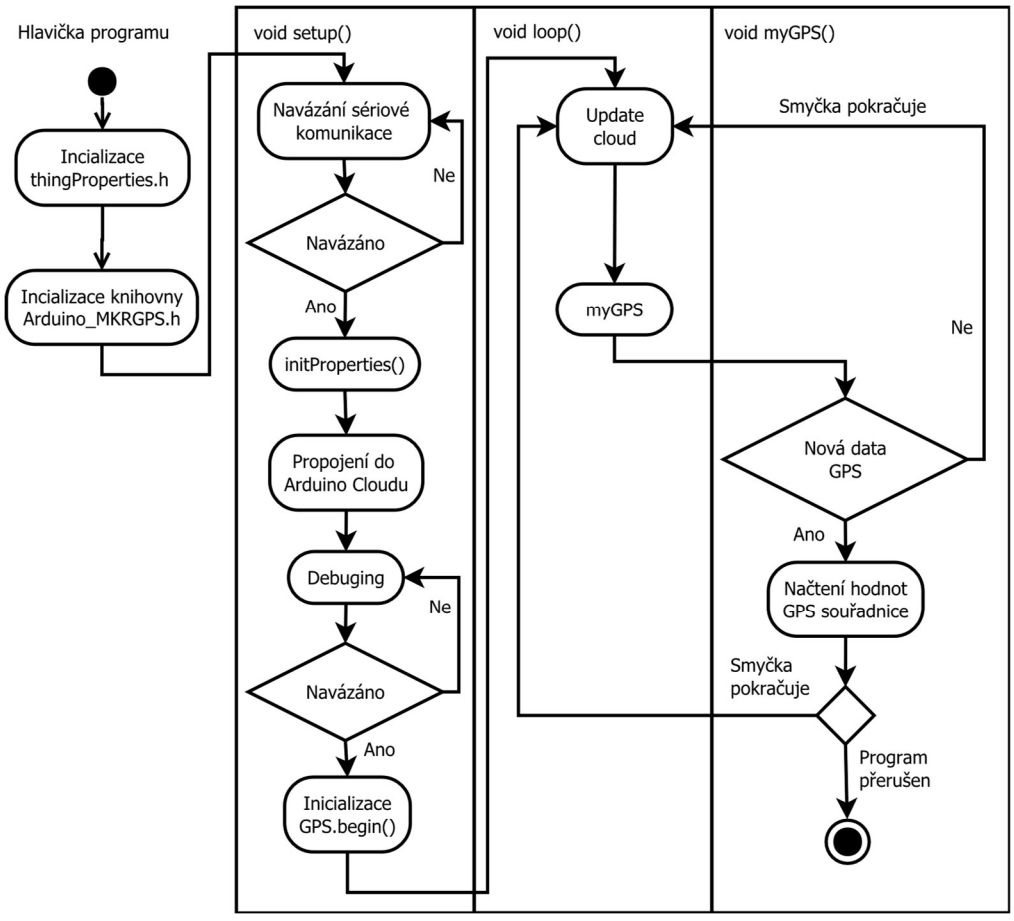
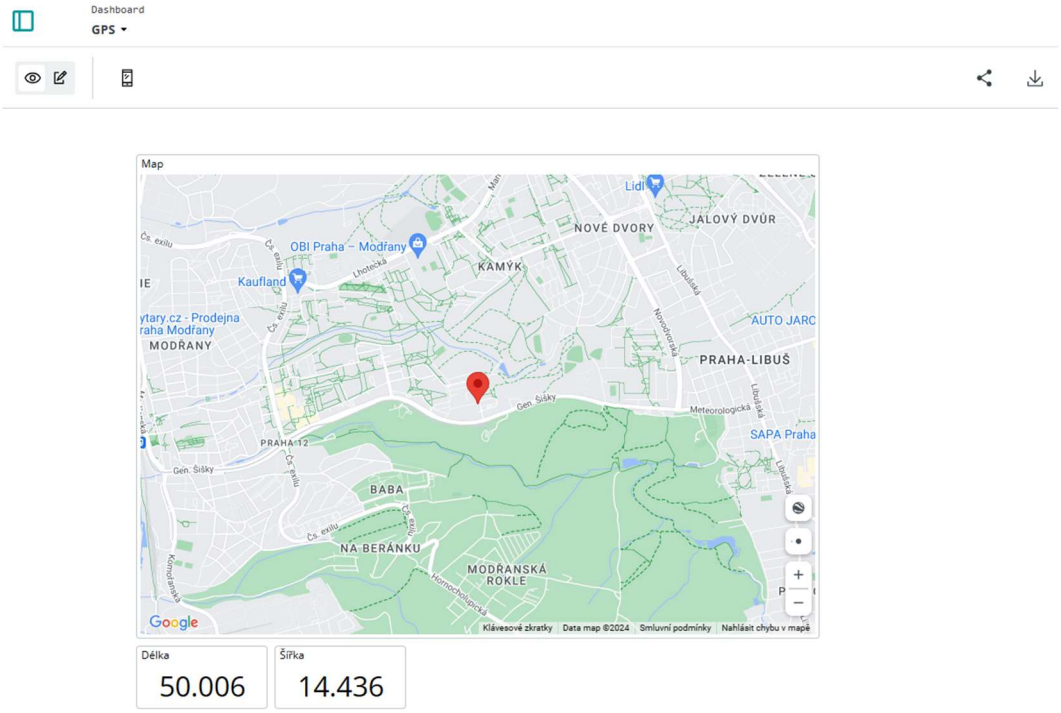



*Diagram 10: GSP*

Jako první definujeme potřebnou knihovnu pro práci s GPS modulem *Arduino_MKRGPS.h*. Zároveň vložíme definici pro Arduino IoT Cloud ze souboru *thingProperties.h*. Jednorázová funkce *void setup()* je podobná jako v předchozích příkladech jen zde navíc inicializujeme GPS modul pomocí metody *GPS.begin()*. Pokračujeme funkcí *void loop()* (nekonečná smyčka) zde pouze voláme *myGPS()* a zároveň aktualizujeme data v cloudu. Z funkce *myGPS()* vyčítáme informace o poloze, rychlosti, nadmořské výšce a počtu přijímaných satelitů. Dále je zde proměnná typu *CloudLocation* s názvem *location*. Pomocí ní aktualizujeme data pro mapu v IoT Cloudu. Z této proměnné načítá mapový widget souřadnice a ty poté zobrazí, tak jak je znázorněno na obrázku číslo 38.



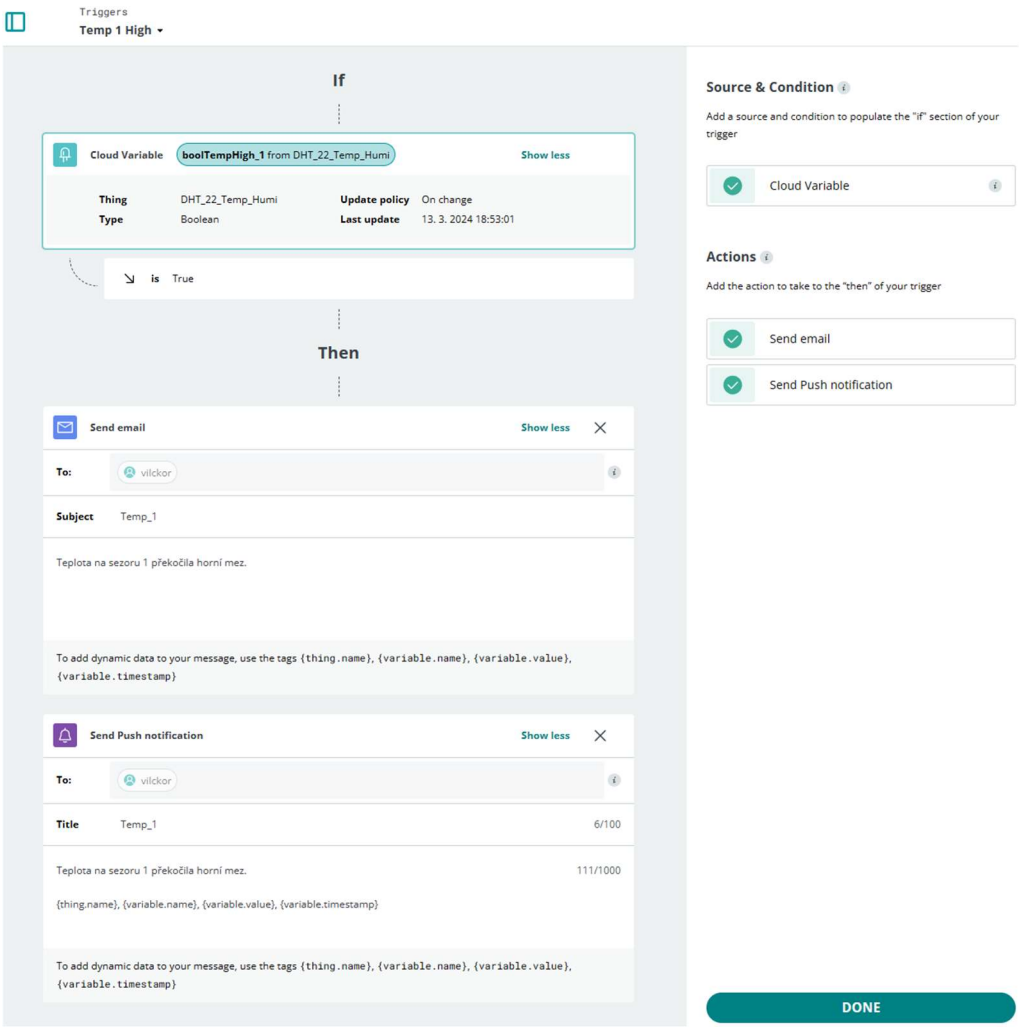

*Obrázek 37: Widget mapy*

# 6 Triggers

Po zapojení a naprogramování tohoto zařízení je potřeba vytvořit takzvaný trigger. Triggery v Arduino Cloudu jsou funkce, které umožňují automatické spouštění akcí nebo upozornění na základě určitých podmínek nebo událostí v projektech Internetu věcí (IoT). Tento systém umožňuje uživatelům definovat pravidla nebo podmínky, při jejichž splnění se automaticky vykoná určitá akce, jako je odeslání upozornění, spuštění skriptu, zapnutí nebo vypnutí zařízení a podobně.

## 6.1 Trigger teploty

V cloudu Arduina na levé straně webu se nachází položka Triggers. Její pomocí vytvoříme snadno například podmínku pro sledování teploty. V případě překroční námi definované meze se automaticky odešle notifikace do aplikace a email s varovnou správou.




*Obrázek 38: Trigger na zasláni notifikace teploty*

V prvním kroku zvolíme zdroj, na který budeme reagovat. V této ukázce je patrné že zdroj bude v projektu *DHT_22_Temp_Humi* sledovaná proměnná je typu Boolean a jmenuje se *boolTemkpHigh_1*. Takto je ve zdrojovém kódu pojmenovaná proměnná, která hlídá překročení teploty na DHT sensoru číslo 1. Ten bude umístěn uvnitř úlu. To samé bylo provedeno pro každý sensor zvlášť, a navíc nebyla hlídána pouze teplota, která byla vyšší než požadovaná ale i ta nízká. Proto na dva základní sensory byly vytvořeny celkem čtyři triggery, které tyto notifikace odesílaly. Na následujícím obrázku je notifikace z aplikace v mobilním telefonu.


*Obrázek 39: Notifikace v mobilní aplikaci*

# 7 Výsledky a diskuse

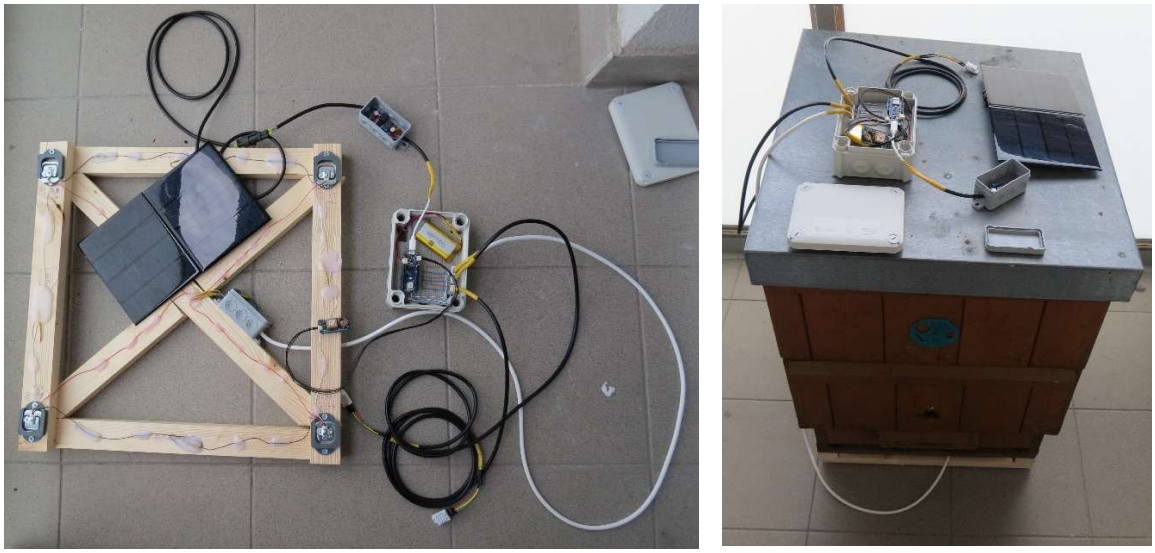
Výsledkem celé práce je hotové a funkční řešení pro monitoring včelího úlu. Na následujících obrázcích je vidět, jakým způsobem je celý projekt řešen. Jen pro rekapitulaci základem je Arduino MKR 1010 WiFi.

Dále je zde jasně patrné, jakým způsobem je zapojeno napájení celé soustavy. V první řadě jsou zde solární panely z důvodů vyššího příkonu pro mikrokontroler a LiPol akumulátor o kapacitě 4000 mAh. Ze solárních panelů je přívodní kabel veden nejprve přes DC/DC měnič napětí a poté je zakončen v mikro USB konektoru.

Další prvek je konstrukce podstavce. Na něm jsou pomocí plastových profilů přichyceny pomocí vrutů tenzometrické senzory. Tyto plastové profily byly vytisknuty na 3D tiskárně přesně na míru. Senzory jsou propojeny přes AD převodník a dále bílím kabelem na konektor Arduina.

Černé kabely jsou zde pro fyzické propojení desky s DHT_22 senzory. Ty, jak bylo zmíněno víše slouží k měření teploty a vzdušné vlhkosti.

A jako poslední je zde vidět GPS Shield MKR ten je připojen přímo na desku k patřičnému konektoru.



*Obrázek 40: Realizace*

## 7.1 Doporučení

Celá sestava funguje velice dobře jen drobnou korekci bych přesto doporučil. Je to pořízení mnohem výkonnějšího solárního panelu. Důvodem je že takto navržené řešení dobře funguje při dobrých slunečních podmínkách. Kdežto při nedostatku světla již příkon nemusí vždy stačit. Doporučil bych panel o výkonu alespoň 20 wattů. Moje řešení se sestává ze dvou panelů, které dohromady dává 6.6 wattů. I toto však přes slunečné dny stačilo jak na chod Arduina, tak na dobytí akumulátoru.

## 7.2 Cenová kalkulace

Jako poslední bude provedena cenová rozvaha, co se týče pořízení jednotlivých prvků a časová náročnost. Většinu komponent jsem koupil na webu drátek.cz. Proto budu vycházet s těchto cen. I když vím že je možné sehnat tyto součástky a za výrazně nižší cenu. Jen je to někdy vykoupeno dražším poštovným, dodacími lhůtami nebo tím že je potřeba odebrat větší množství což nebylo mým cílem.

*Tabulka 3: Rozpočet součástky*

| **Položka** | **Kus (kč)** | **Celkem (kč)** | **Dodavatel** |
|---|---|---|---|
| Sada JST konektorů samice + samec 2, 3, 4, 5Pin 2,54 mm | 75 | 75 | dratek.cz |
| Váhový senzor - 50 kg | 25 | 100 | dratek.cz |
| DHT22 digitální teploměr a vlhkoměr | 114 | 228 | dratek.cz |
| AD Převodník Modul 24-bit 2 kanály HX711 | 30 | 30 | dratek.cz |
| Odolné smršťující trubice 45 mm v boxu - 452 kusů | 290 | 290 | dratek.cz |
| Solární panel 6V 3,3W až 550mA | 219 | 438 | dratek.cz |
| Boost-buck step up/down modul solárního napájení DC-DC XL6009 | 67 | 67 | dratek.cz |
| Arduino MKR GPS Shield | 870 | 870 | store.arduino.cc |
| Arduino MKR WiFi 1010 | 838 | 838 | store.arduino.cc |
| Dřevo na konstrukci váhy | 150 | 150 | Hornbach |
| Kabely | 100 | 100 | Hornbach |
| Pájivé kontaktní pole | 40 | 40 | |
| OBO Rozbočovací krabice IP66 | 100 | 100 | Obi |
| Rozbočovací krabice na omítku 75 mm x 45 mm, šedá, IP54 | 36 | 72 | Obi |
| GeB LiPol Baterie 785084 4000mAh 3.7V JST-PH 2.0 | 318 | 318 | laskakit.cz |
| **Celková cena za materiál** | | **3716 kč** | |

Cena zahrnuje jen materiál, ne zahrnuje práci ani energii. Kromě toho, pokud by se vybrala jiná deska s odlišným typem konektivity, například Arduino MKR WAN 1300 za 1050 Kč, nezvýší se tím cena nejvíce. Avšak přidání WisGate Edge Lite 2 od Arduino za 5400 Kč by vyžadovalo další investici. Další možností by byla volba paušálu od operátora, kdy například České radiokomunikace nabízejí balíček za cenu 200 Kč pro 10 IoT zařízení. Tato cena byla relevantní v době psaní této diplomové práce.

# 8 Závěr

Cíle této diplomové práce byly splněny. Během několika dní byl sestrojen prototyp zařízení pro monitoring stavu včelstva. Konstrukce byla provedena z běžně dostupných komponent, které se dají pořídit za přijatelné ceny na internetu. Prvních několik dní probíhalo seznamování s ekosystémem Arduina. Jednalo se o programování zvoleného mikrokontroleru. Pak práci v jeho cloudu, kdy jsem celé zařízení naprogramoval a s ním propojil. Vše prve vznikalo za pomoci nepájivého pole, kdy jsem jednotlivé prvky mohl snadno připojovat a odpojovat. Tím operativně měnit hardwarovou konfiguraci a zapojení. Použity byly i další prvky, které jsem do této práce již nezahrnoval. Byly to například krokové motory, hardwarová tlačítka, různé displeje, vibrační snímač a jiné.

Po čase, kdy se ustálila představa, jak by celé zařízení mělo vypadat v podobě prvního prototypu bylo přistoupeno ke kompletaci. Celá sestava je propojena již prostřednictvím pájivého pole. To je poté vloženo do plastové elektrikářské krabičky, která má krytí IP 66. Ta nám zajišťuje úplné krytí proti prachu a silným proudům vody. Konektory byly též voleny s ohledem na to, že zařízení bude nepřetržitě fungovat venku a musí tedy odolat našim klimatickým podmínkám.

V další fázi by již bylo přistoupeno ke zdokonalování sestavy. Pro nasazení by byl vyleptán plošný spoj. Dále by byly testovány různé typy kabelů a konektorů, aby bylo docíleno robustnosti a odolnosti zařízení pro celoroční nasazení venku. Samozřejmě podstava pod váhu by prošla jistě také designovou změnou.

Nicméně i takto postavený prototyp nepřetržitě venku fungoval několik týdnů bez nejmenších problémů. Práce s deskou Arduino se nesmírně osvědčila její výborná kvalita zpracování a podpora je ideální pro rychlí vývoj a nasazení do produkce. Výborně se na ní učí jednotlivé principy práce s různými typy periferií. Sestavení kódu je z pravidla velice jednoduché a intuitivní. Dle mého názoru ji vřele doporučuji. Jsem si jistý že tuto platformu budu i nadále využívat.

# 9 Seznam použitých zdrojů

[1] ARDUINO. MKR WAN 1300. ARDUINO. Arduino [online]. 2023 [cit. 2023-11-16]. Dostupné z: https://docs.arduino.cc/hardware/mkr-wan-1300

[2] LORA. What are LoRa® and LoRaWAN®? [online]. December 2019 [cit. 2024-01-15]. Dostupné z: https://lora-developers.semtech.com/uploads/documents/files/LoRa_and_LoRaWAN-A_Tech_Overview-Downloadable.pdf

[3] LORA ALLIANCE®. What is LoRaWAN® Specification [online]. 2023 [cit. 2023-11-16]. Dostupné z: https://lora-alliance.org/about-lorawan/

[4] POSPÍŠIL, Ondřej a Radek FUJDIAK. FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ VUT V BRNĚ. Ověření Bezpečnosti LoRaWAN Sítě Pomocí Simulovaných Scénářů. Http://trilobit.fai.utb.cz/ [online]. 2020, 1.6.2020, 1.6.2020(1), 12 [cit. 2023-11-16]. Dostupné z: http://trilobit.fai.utb.cz/Data/Articles/PDF/3ab7986b-f1fb-4602-a4d8-3c3fc755246f.pdf

[5] FORBES. Konec 3G v Čechách. Operátoři letos starší sítě vypnou [online]. 2021 [cit. 2023-11-17]. Dostupné z: https://forbes.cz/konec-3g-v-cechach-operatori-letos-starsi-site-vypnou/#:~:text=%C4%8CTK%20Po%2017%20letech%20letos,si%20zm%C4%9Bny%20technologie%20ani%20nev%C5%A1imne

[6] PRAVDA, Ivan. Mobilní a bezdrátové sítě [online]. 1. České vysoké učení technické v Praze, 2015 [cit. 2023-11-17]. Dostupné z: https://publi.cz/books/236/Cover.html

[7] KRČMÁŘ, Petr. Mobilní sítě 2G tu s námi budou minimálně do roku 2028. Root.cz [online]. 2021, 16. 12. 2021 [cit. 2023-11-17]. Dostupné z: https://www.root.cz/zpravicky/mobilni-site-2g-tu-s-nami-budou-minimalne-do-roku-2028/#:~:text=%C4%8CT%C3%9A%20nyn%C3%AD%20obnovil%20p%C5%99%C3%ADd%C4%9Bl%20p%C3%A1sma,P%C5%99edsedkyn%C4%9B%20Rady%20%C4%8CT%C3%9A%20Hana

[8] ARDUINO. MKR GSM 1400 [online]. 2023 [cit. 2023-11-17]. Dostupné z: https://docs.arduino.cc/hardware/mkr-gsm-1400

[9] ARDUINO. MKR NB 1500. Https://www.arduino.cc/ [online]. 2023 [cit. 2023-11-18]. Dostupné z: https://store.arduino.cc/products/arduino-mkr-nb-1500

[10] Tutorial LTE [online]. 2023 [cit. 2023-11-18]. Dostupné z: https://www.tutorialspoint.com/lte/lte_network_architecture.htm

[11] ERICSSON. Know the difference between NB-IoT vs. Cat-M1 for your massive IoT deployment [online]. 2019 [cit. 2023-11-18]. Dostupné z: https://www.ericsson.com/en/blog/2019/2/difference-between-nb-iot-cat-m1

[12] ARDUINO. Arduino MKR WiFi 1010. ARDUINO. Arduino Store [online]. 2021 [cit. 2023-11-22]. Dostupné z: https://store.arduino.cc/products/arduino-mkr-wifi-1010

[13] Power Tree. In: Arduino [online]. 2024 [cit. 2024-01-03]. Dostupné z: https://docs.arduino.cc/resources/datasheets/ABX00023-datasheet.pdf

[14] Pinout diagram MKR WiFi 1010. In: ARDUINO. Store Arduino [online]. 2020, 7.8.2020, 7.8.2020 [cit. 2024-01-03]. Dostupné z: https://content.arduino.cc/assets/Pinout-MKRwifi1010_latest.pdf

[15] Sensor 50Kg. In: Https://dratek.cz/ [online]. 2024 [cit. 2024-01-06]. Dostupné z: https://dratek.cz/arduino/2202-vahovy-senzor-50-kg.html

[16] AVIA SEMICONDUCTORS. 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales. AVIA SEMICONDUCTORS. Alldatasheet.com [online]. 2018, 2. 7. 2018, 2. 4. 2019 [cit. 2024-01-07]. Dostupné z: https://pdf1.alldatasheet.com/datasheet-pdf/view/1132222/AVIA/HX711.html

[17] AOSONG ELECTRONICS CO.,LTD. Alldatasheet.com. AOSONG ELECTRONICS CO.,LTD. Alldatasheet.com [online]. 2010, 28. 9. 2010, 5. 4. 2019 [cit. 2024-01-07]. Dostupné z: https://pdf1.alldatasheet.com/datasheet-pdf/download/1132459/ETC2/DHT22.html

[18] MICROCONTROLLER TUTORIALS AND RESOURCES. How the DHT22 Sensor Works. Teachmemicro.com [online]. 2019, 2.2.2019 [cit. 2024-01-09]. Dostupné z: https://www.teachmemicro.com/how-dht22-sensor-works/

[19] DRÁTEK. Solární panel 6V 3,3W až 550mA [online]. 2023 [cit. 2024-02-23]. Dostupné z: https://dratek.cz/arduino/121319-solarni-panel-6v-3-3w-az-550ma.html

[20] DRÁTEK. Boost-buck step up/down modul solárního napájení-nastavitelný DC-DC XL6009 [online]. 2023 [cit. 2024-02-23]. Dostupné z: https://dratek.cz/arduino/1752-boost-buck-step-up-down-modul-solarniho-napajeni-nastavitelny-dc-dc-xl6009.html

[21] COMPONENTS101. XL6009 PWM Switching Regulator (Buck-Boost) [online]. 2019 [cit. 2024-02-23]. Dostupné z: https://components101.com/regulators/xl6009-pwm-switching-regulator-buck-boost

[22] ARDUINO. Arduino MKR GPS Shield [online]. 2021 [cit. 2024-02-13]. Dostupné z: https://store.arduino.cc/products/arduino-mkr-gps-shield

[23] ARDUINO. Arduino Documentation [online]. 2024 [cit. 2024-02-13]. Dostupné z: https://docs.arduino.cc/

[24] JST. PH conector 2.0 mm pitch/Disconatable Crimp style connectors [online]. 2024 [cit. 2024-03-11]. Dostupné z: https://www.jst.com/wp-content/uploads/2021/01/ePH-H.pdf

# 10 Přílohy

Příloha A Zdrojový kód

Příloha B Soubor thingProperties.h

# Příloha A Zdrojový kód

```
/*
  Sketch generated by the Arduino IoT Cloud Thing "Full code"
  https://create.arduino.cc/cloud/things/7d29d1de-bb46-401c-a921-74ab18a96dcc

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when changes are made to the Thing

  String kiloString;
  float altitude;
  float gram;
  float kilo;
  float latitude;
  float longitude;
  float speed;
  float temp_1;
  float temp_2;
  int calibrationWeight;
  int humi_1;
  int humi_2;
  int satellites;
  int setTempHigh_1;
  int setTempHigh_2;
  int setTempLow_1;
  int setTempLow_2;
  CloudLocation location;
  bool boolTempHigh_1;
  bool boolTempHigh_2;
  bool boolTempLow_1;
  bool boolTempLow_2;
  bool buttonCalibration;
```

```
  bool buttonTare;
  bool ledCalibration;
  bool ledTare;

  Variables which are marked as READ/WRITE in the Cloud Thing will also have functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this sketch.
*/
/*****************************************************************************
***************/

#include "thingProperties.h"
#include "HX711.h"
#include "DHT.h"
#include <Arduino_MKRGPS.h>

/*********************************/
/*Scale declaration*/

HX711 scale;

uint8_t dataPin = 6;
uint8_t clockPin = 7;

float measuredWeightValue = 0;
float zeroWeightUnits = 0;
float calibratinWeightUnits = 0;
float gramsUnit = 0;

/*End scale declaration*/

/*********************************/
```

```
/*DHT22 declaration*/

#define pinDHT1 4
#define pinDHT2 5

#define typDHT22 DHT22

/*Instance sensorů*/
DHT DHT1(pinDHT1, typDHT22);
DHT DHT2(pinDHT2, typDHT22);

/*End DHT22 declaration*/

/**************************************************************************
***************/

void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if none is found
  delay(1500);

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  scale.begin(dataPin, clockPin); //

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
```

```
    the higher number the more granular information you'll get.
    The default is 0 (only errors).
    Maximum is 4
 */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

  if (!GPS.begin()) {
    Serial.println("Failed to initialize GPS!");
    while (1);
  }
}

/*****************************************************************************
***************/

/*Begin Loop*/
void loop() {
  ArduinoCloud.update();
  // Your code here
  myScale();
  myDHT();
  myTrigerTempHigh();
  myTrigerTempLow();
  myGPS();

}
/*End Loop*/

/*****************************************************************************
***************/

/*Begin Scale*/
```

```
void myScale () {
  measuredWeightValue = scale.get_units(10);  // read until stable
  delay(1000);

  gram = measuredWeightValue/gramsUnit;   //gram unit
  kilo = gram/1000;                       // Kg unit
  kiloString = String(kilo, 3);           //Retipe and rounding

  Serial.print("TRANSFER VALUE: ");       //Gram unit for calibration
  Serial.println(calibrationWeight);

  Serial.print("MEASURE VALUE: ");        //measure unit from hx711
  Serial.println(measuredWeightValue);

  Serial.print("GRAMS UNITS: ");          //Number of units per gram
  Serial.print(gramsUnit);
  Serial.println(" Grams");

  Serial.print(kiloString);               //String kg
  Serial.println(" KG String");

  delay(5000);
}
/*End Scale*/

/*
  Since ButtonCalibration is READ_WRITE variable, onButtonCalibrationChange() is
  executed every time a new value is received from IoT Cloud.
  This function set up calibratin
*/
void onButtonCalibrationChange()  {

  if(buttonCalibration == HIGH){
```

```
    gramsUnit = measuredWeightValue/calibrationWeight;    //set units per gram
    ledCalibration = HIGH;
   }
  else{
    ledCalibration = LOW;
   }
}

/*
  Since ButtonTare is READ_WRITE variable, onButtonTareChange() is
  executed every time a new value is received from IoT Cloud.
  This is Tare function. Set up zero for measuremet value.
*/
void onButtonTareChange()  {

    if(buttonTare == HIGH){
    scale.tare();                  //Tare function
    ledTare = HIGH;
  }
  else{
    //Serial.println("LOW TARA");
    ledTare = LOW;
  }
}

/*
  Since CalibrationWeight is READ_WRITE variable, onCalibrationWeightChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onCalibrationWeightChange()  {
}
```

```
/**************************************************************************
**************/

/*DHT blok*/
/*
   Reads humidity and temperature values from both DHT22 sensors.
 */
 void myDHT() {
   temp_1=DHT1.readTemperature();
   temp_2=DHT2.readTemperature();

   humi_1=DHT1.readHumidity();
   humi_2=DHT2.readHumidity();
 }

// Triger High temp
void myTrigerTempHigh() {
   if(temp_1>setTempHigh_1)
   {
     boolTempHigh_1 = true;
   }
   else
   {
     boolTempHigh_1 = false;
   }

 if(temp_2>setTempHigh_2)
   {
     boolTempHigh_2 = true;
   }
   else
   {
     boolTempHigh_2 = false;
```

```
    }
  }

// Triger low temp
void myTrigerTempLow() {
    if(temp_1<setTempLow_1)
    {
      boolTempLow_1 = true;
    }
    else
    {
      boolTempLow_1 = false;
    }

  if(temp_2<setTempLow_2)
    {
      boolTempHigh_2 = true;
    }
    else
    {
      boolTempHigh_2 = false;
    }
  }

/*
  Since SetTempHigh1 is READ_WRITE variable, onSetTempHigh1Change() is
  executed every time a new value is received from IoT Cloud.
*/
void onSetTempHigh1Change()  {
  // Add your code here to act upon SetTempHigh1 change
}

/*
```

```
  Since SetTempHigh2 is READ_WRITE variable, onSetTempHigh2Change() is
  executed every time a new value is received from IoT Cloud.
*/
void onSetTempHigh2Change()  {
  // Add your code here to act upon SetTempHigh2 change
}

/*
  Since SetTempLow1 is READ_WRITE variable, onSetTempLow1Change() is
  executed every time a new value is received from IoT Cloud.
*/
void onSetTempLow1Change()  {
  // Add your code here to act upon SetTempLow1 change
}

/*
  Since SetTempLow2 is READ_WRITE variable, onSetTempLow2Change() is
  executed every time a new value is received from IoT Cloud.
*/
void onSetTempLow2Change()  {
  // Add your code here to act upon SetTempLow2 change
}
/*End DHT blok*/

/*****************************************************************************
***************/

/*Blok GPS*/

void myGPS() {
  // check if there is new GPS data available

  if (GPS.available()) {
```

```
    // read GPS values
    float latitude   = GPS.latitude();
    float longitude  = GPS.longitude();
    float altitude   = GPS.altitude();
    float speed      = GPS.speed();
    int   satellites = GPS.satellites();
    location = {latitude, longitude};
    }
}

/*End Blok GPS*/
```

# Příloha B Soubor thingProperties.h

```
// Code generated by Arduino IoT Cloud, DO NOT EDIT.

#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>

const char SSID[]     = SECRET_SSID;    // Network SSID (name)
const char PASS[]     = SECRET_OPTIONAL_PASS;    // Network password (use for WPA,
or use as key for WEP)

void onCalibrationWeightChange();
void onSetTempHigh1Change();
void onSetTempHigh2Change();
void onSetTempLow1Change();
void onSetTempLow2Change();
void onButtonCalibrationChange();
void onButtonTareChange();

String kiloString;
float altitude;
float gram;
float kilo;
float latitude;
float longitude;
float speed;
float temp_1;
float temp_2;
int calibrationWeight;
int humi_1;
int humi_2;
int satellites;
int setTempHigh_1;
int setTempHigh_2;
```

```
int setTempLow_1;
int setTempLow_2;
CloudLocation location;
bool boolTempHigh_1;
bool boolTempHigh_2;
bool boolTempLow_1;
bool boolTempLow_2;
bool buttonCalibration;
bool buttonTare;
bool ledCalibration;
bool ledTare;

void initProperties(){

  ArduinoCloud.addProperty(kiloString, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(altitude, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(gram, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(kilo, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(latitude, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(longitude, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(speed, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(temp_1, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(temp_2, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(calibrationWeight, READWRITE, ON_CHANGE,
onCalibrationWeightChange);
  ArduinoCloud.addProperty(humi_1, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(humi_2, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(satellites, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(setTempHigh_1, READWRITE, ON_CHANGE,
onSetTempHigh1Change);
  ArduinoCloud.addProperty(setTempHigh_2, READWRITE, ON_CHANGE,
onSetTempHigh2Change);
```

```
  ArduinoCloud.addProperty(setTempLow_1, READWRITE, ON_CHANGE,
onSetTempLow1Change);
  ArduinoCloud.addProperty(setTempLow_2, READWRITE, ON_CHANGE,
onSetTempLow2Change);
  ArduinoCloud.addProperty(location, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(boolTempHigh_1, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(boolTempHigh_2, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(boolTempLow_1, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(boolTempLow_2, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(buttonCalibration, READWRITE, ON_CHANGE,
onButtonCalibrationChange);
  ArduinoCloud.addProperty(buttonTare, READWRITE, ON_CHANGE,
onButtonTareChange);
  ArduinoCloud.addProperty(ledCalibration, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(ledTare, READ, ON_CHANGE, NULL);

}

WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```