NEW MEDIA &
COMMUNICATION
TECHNOLOGY

Verslag project Datacommunicatie

Sodaq mbili. (Arduino + raspberry pi)

howest
De Hogeschool West-Vlaanderen
Lid van de Associatie Universiteit Gent

Opdracht

We hebben voor opdracht 10 gekozen. Dit is de opdracht over Sodaq mbili (Arduino + raspberry pi). Hierbij werd er gevraagd om sensoren aan te sluiten op de Sodaq mbili en deze waarden door te sturen naar een raspberry pi over seriële communicatie . Daar moet het opgeslagen worden op een database. Deze waarden kunnen dan geraadpleegd worden op een website. Bovendien moeten alle aansturingen op de Arduino gebeuren zoals licht aan steken als het donker wordt.

Arduino/mbili

De sodaq mbili (wat 2 betekent) is de opvolger van de succesvolle sodaq one. Het is een Arduino platform die het makkelijk maakt om verschillende sensoren en actuatoren aan te sluiten.

De sodaq borden zijn speciaal ontwikkeld voor IoT. Dit is zo klein mogelijk gemaakt om in alle mogelijke toepassingen te passen. Bovendien kan de Mbili makkelijk uitgerust worden met een zonnepaneel en batterij zodat die niet moet aangesloten zijn aan het elektriciteitsnet.

De sodaq mbili bezit volgende features

- 6 pin ICSP header
- Een switched grove row
- Een always on grove row
- 2 programeerbare leds
- ATMega 1284p microcontroller
- Usb poort (stroom en data)
- Batterij poort
- Zonnepaneel poort

Om de sodaq mbili te programmeren hebben we de laatste versie nodig van de arduino IDE. Deze kan zowel op windows als op de raspberry pi gedraaid worden via pixel. Deze software kan gedownload worden op de Arduino website.

Vervolgens moeten we ook nog de SODAQ library installeren op de arduino IDE. Dit kan gedaan worden via de ingebouwde library manager. Eenmaal de library geïnstalleerd is moet je nog het correctie SODAQ bord kiezen en de correcte USB poort waarop de SODAQ is aangesloten.

De mbili kit komt ook met een uitgebreid assortiment aan sensoren. We gaan deze niet allemaal gebruiken. Wij gebruiken volgende sensoren.

- Lichtsensor
- luchtvervuiling
- Temperatuur
- Luchtdruk
- Luchtvochtigheid
- GPS

Alle data die we binnenkrijgen van de sensor zullen we formateren en naar onze node doorsturen via de seriële poort.

C++

Het sodaq mbili bord word geprogrammeerd in de C++ taal die de opvolger is van de C taal. De naam C++ is afkomstig van de programma opdracht waarbij ++ betekent dat je de waarde van 1 verhoogt. C++ is ook object georiënteerd.

C++ is ontworpen om gebruikt te worden voor het programmeren van verschillende systemen en embedded systemen. Het word ook veel gebruikt voor het ontwikkelen van desktopapplicaties.

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

TinyGPSPlus tinyGPS; // Create a TinyGPSPlus object
SoftwareSerial gpsPort(2, 1);
Adafruit_BME280 bme; // I2C
int luchtVervuilingSensor = A0;
float lat,lng;
int lichtSensor = A4;
int geluidSensor = A2;

void setup() {
  pinMode(LED1, OUTPUT);
  digitalWrite(LED1, LOW);
  // put your setup code here, to run once:
  Serial.begin(9600);
  gpsPort.begin(9600);
  if (!bme.begin()) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  };
}

void loop() {

  printGPSInfo();
  smartDelay(1000);

    }

static void smartDelay(unsigned long ms)
{
  unsigned long start = millis();
  do
  {
    // If data has come in from the GPS module
    while (gpsPort.available())
      tinyGPS.encode(gpsPort.read()); // Send it to the encode function
    // tinyGPS.encode(char) continues to "load" the tinGPS object with new
    // data coming in from the GPS module. As full NMEA strings begin to come in
    // the tinyGPS library will be able to start parsing them for pertinent info
  } while (millis() - start < ms);
}
```

```
void printGPSInfo()
{
  // Print latitude, longitude, altitude in feet, course, speed, date, time,
  // and the number of visible satellites.
    lat = tinyGPS.location.lat()*1000;
    lng = tinyGPS.location.lng()*1000;

       Serial.println("LATITUDE:" + (String)lat + "#");
  Serial.println("LONGITUDE:" + (String)lng + "#");
  Serial.println("licht:" + (String)analogRead(lichtSensor) + "#");
           Serial.println("lucht:" + (String)analogRead(luchtVervuilingSensor) + "#");
           Serial.println("temp:"  + (String)bme.readTemperature() + "#");
           Serial.println("press:" + (String)(bme.readPressure() / 100.0F) + "#");
           Serial.println("vocht:" + (String)bme.readHumidity() + "#");


      if (analogRead(lichtSensor) > 150){
        digitalWrite(LED1, LOW);
      }else{
          digitalWrite(LED1, HIGH);
        }

          |
}
```

Een korte verduidelijking van deze code.

- In de setup functie worden de seriële poort ingesteld voor communicatie, de gps initialiseren en de bme sensor activeren.
- In de loop functie printen we de gps info en roepen we de smartdelay op.
- De smartdelay functie zal het systeem doen pauzeren terwijl het toch nog verschillende dingen kan laden op de achtergrond
- PrintGPSInfo zal de lat, lng van uw locatie ophalen en alle sensoren uitlezen en meteen verzenden via de seriële poort
- Als afwerking zullen we een van de ledjes op het bord doen oplichten wanneer de lichtwaarde onder een bepaalde waarde komt.

Raspberry pi

De raspberry pi is ontwikkeld voor schoolomgevingen omdat het erg gemakkelijk is om mee te experimenteren. Het volledige OS-systeem en alle bestanden staan op de SD-kaart. Als je bijvoorbeeld aan een ander project wilt werken kan u gewoon de SD-kaart wisselen en hoeft u niet telkens alles te overschrijven. Door deze functionaliteit word de raspberry pi ook veel gebruikt bij IoT toepassingen of bij prototyping.

Er zijn reeds meerdere versies beschikbaar van de raspberry pi, wij gebruiken de raspberry pi 3. Enkele features van dit model zijn:

- Wifi
- Bluetooth
- 1GB RAM,
- 4 USB poorten
- 40 GPIO pins
- Ethernet poort

- Full HDMI poort
- Combined 3.5mm audio jack
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- videoCore IV 3D graphics slot

Naast de raspberry pi heb je enkele zaken nodig om hem te laten werken. Wij gebruiken een stroombron van 2A, een 16GB SD-kaart een ethernet kabel. Om ervoor te zorgen dat dit project goed werkt moet de raspberry pi verbonden zijn met een werkende internet connectie. Wanneer het weerstation zich in een lokaal netwerk bevindt moet er aan port-forwarding gedaan worden om de data te kunnen ophalen van buitenaf. Voor de api en website gebruiken we poort 80 en voor de mysql gebruiken we 3306.

Voor de raspberry pi de eerste keer op te zetten heb je enkele benodigdheden nodig. Voor de eerste setup heb je ook een beeldscherm en een toetsenbord nodig. Eenmaal de raspberry pi geïnstalleerd is kunnen we via ssh of vnc van een remote computer inloggen op de raspberry pi en die van daaruit besturen.

Voor de OS op de raspberry pi zullen we de laatste versie van “Raspbian jessie with pixel“ gebruiken. Raspbian is een linux OS en heeft al heel wat zaken voorhand geïnstalleerd zoals python, een visueel bureaublad en verschillende protocollen. Om de OS op de SD-kaart te installeren gebruiken we op onze pc het “Win32DiskImages” programma en een SD-kaartlezer.

Seriële communicatie

Bij seriële communicatie worden bits 1 voor 1 asynchroon verstuurt over een enkele draad. Alle informatie die verstuurd wordt zal ontvangen worden in een buffer. Telkens als je deze buffer uitleest zal deze geleegd worden. Uit ervaring ondervinden we data die mooi in 1 woord verstuurt word niet altijd in 1 mooi woord aankomt maar in verschillende segmenten is opgedeeld. Daarom is het goed om met een eigen “stopteken” te werken. Je zal de data blijven uitlezen en aan elkaar plakken totdat je het stopteken tegenkomt. Alles daarvoor was dus 1 bericht.

We maken gebruik van een eigen protocol met een “stopteken” om de verschillende waarden van elkaar te onderscheiden. Voor het protocol zullen we eerst een uniek keyword versturen die aanduid welke waarde je zal binnenkrijgen. Vervolgens word een “:” gestuurt om het keyword en de data van elkaar te splitsen. Na de dubbele punt word de data verstuurt en word er een “#” geplaatst. De “#” duid aan dat dit het einde van het bericht is en dat er opnieuw naar een nieuw keyword kan geluisterd worden.

We zullen dus met dit protocol de data verzenden vanuit de arduino op de seriële poort. Deze zal ontvangen worden bij de raspberry pi

Het is wel belangrijk dat beide partijen (zender en ontvanger) lezen en schrijven aan de zelfde snelheid. Deze snelheid is uitgedrukt in baud. Als zender en ontvanger niet aan dezelfde snelheid lezen en schrijven zullen bits verloren gaan en zal de ontvanger enkel onzin uitlezen.

Keuze technologieën

Naast de Sodaq Mbili en Raspberry pi waar we onze opdracht mee realiseerden hebben we ook enkele andere technologieën gebruikt.

Database

Als database kozen we voor mysql. Dit is een snelle en efficiënte relationele database. De andere mogelijkheid die we overwogen hebben is MongoDB. Dit is een NOSQL ,dit betekend niet relationeel. Deze database werkt volledig met JSON zodat deze niet meer moet omgevormd worden voor de

server het naar de front end stuurt. Dit resulteert dus in snelheidswinst. Desondanks de kracht en snelheid van MongoDB hebben we er toch voor gekozen om voor de meer klassieke Mysql te gaan.

Mysql

Mysql is een database managementsysteem dat de SQL taal gebruikt om data te bekijken, toevoegen en manipuleren. Mysql word vooral gebruikt voor internettoepassingen meestal in combinatie met php. Wij zullen echter geen php gebruiken maar dit realiseren met Node.js.

Het linux pakket voor mysql word ook wel mysqld genoemd, De “D” staat voor daemon. Daemon is een term in linux voor een process dat op een computer draait zonder inmenging van de gebruiker.

Om mysql te installeren op onze raspberry pi gebruiken we het commando “sudo apt-get install mysqld”. We kunnen ons programma beheren in de mysql shell, deze kan je activeren met het commando “sudo mysql -u root -p”. van hieruit kan ja met de sql taal uw verschillende databases en tables beheren.

Server + seriële communicatie

Voor server en seriële communicatie waren er twee opties.

De eerste mogelijkheid die we overwogen hebben is om met python te werken voor de seriële communicatie en Apache met php als webserver. Dit grotendeels omdat dit de technologie is die ons door NMCT is aangeleerd.

De tweede mogelijkheid is Node.js. We hebben gekozen voor deze optie aangezien dit zowel de taken van webserver kan uitvoeren als de taken van een taal zoals python die de seriële poorten uit kan lezen.

Node.js

Node.js is een server-sided platform dat gebaseerd is op de V8 engine van Google Chrome's Javascript Engine. Node.js is een open-source, cross-platform runtime environment. Dit wil zeggen dat server-sided en network applications hierop kunnen worden gemaakt. Node.js applicaties zijn geschreven in Javascript. Iedereen kent JavaScript, Het is een scripting-taal die meestal gebruikt wordt om interactieve websites te maken.

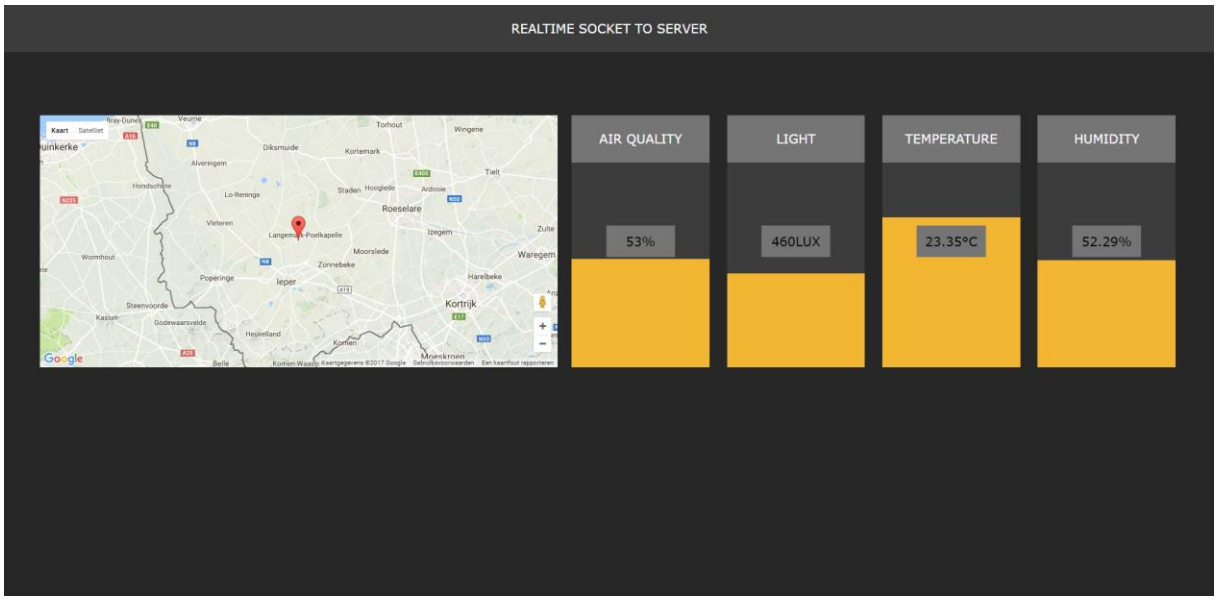
- Asynchronous and Event driven Alle API's en Node.js Library's zijn asynchroon. Dit betekent dat Node.js servers nooit wachten op een API of database die data hoort terug te sturen. De response van de request wordt later opgevraagt door de Events mechanisme.
- Zeer snel Node.js Library is gebouwd op Google Chrome's V8 Javascript Engine en is zeer snel in het uitvoeren van code. Single Threaded but Highly Scalable Node.js gebruikt een 1 threaded model samen met event looping. Dit mechanisme helpt de server bij het terugsturen van responses in een non-blocking way (Node wacht nooit op een response.) Dit zorgt er dus voor dat er zeer veel tergelijkertijd kan gedaan worden tegenover een traditionele server die moet wachten op threads die een response terugsturen.

Express is een minimaal en flexibele Node.js web applicatie framework die een robuuste set van features voor web en mobile apps kan aanbieden. De snelheid van express wordt gegarandeerd door een dunne laag van fundamentele web applicaties en features zonder dat de features en kenmerken van Node.js in de weg zitten.

Website

We hebben een website gebouwd die live data toont van de sensoren. Zo kunnen we live zien hoeveel licht er aanwezig is in de ruimte. Deze website toont ook de huidige locatie van de Arduino.

Deze website toont live de data via socket.io van de server.



Indien je alle data wil bekijken die bijgehouden is kan je kijken op het /api/ pad van de webserver. De data die de webserver stuurt is json. Daarmee kan je makkelijk rapporten opstellen met verschillende tools. Indien je enkel de laatste data wil bekijken buiten de website kan dit ook vie het pad /api/latest.

planning

Onderdeel	**Persoon**	**Tijd**
Onderzoek Componenten en applicaties	Brecht Valcke en Siebe Verschaeve	6u
Opzetten OS op raspberry pi	Brecht Valcke en Siebe Verschaeve	2u
Node basisstructuur opzetten	Brecht Valcke	30min
Node Express instellen	Brecht Valcke	30min
Arduino script schrijven	Siebe Verschaeve	2,5u
Node API's	Siebe Verschaeve	30min
Node Seriële poort uit lezen	Brecht Valcke	1,5u
Node Socket IO	Brecht Valcke	2u
Client side Socket IO	Brecht Valcke	15min
Front end scripting en styling	Siebe Verschaeve	30min
Mysql database opstellen en beheren	Siebe Verschaeve	30min
Google maps integratie	Siebe Verschaeve	30min

Testing	Siebe Verschaeve en Brecht Valcke	4u

Samenvatting

Het project was zeer interessant. We leerden hoe seriële communicatie werkte en dit lag dus in perfecte lijn met wat deze module ons moet leren. Het beste aan de opdracht is dat het zeer breed was. We hebben verschillende zaken gerealiseerd met verschillende technologieën. Dit ging niet enkel over de communicatie maar ook de programmatie rond de communicatie. De data heeft al een hele weg afgelegd voor het op de website weergegeven wordt. Dit is dus een ideaal voorbeeld van hoe je data van 1 plaats naar de andere brengt en dit volledig opslaat onderweg. Deze opdracht gaf ons ook de kans om met technologieën te werken die we nog nooit gebruikt hadden.

Git repository: https://github.com/verschaevesiebe/Datacommunication