

Jarkko Aho

# OBD-etälukulaite

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Innovaatioprojekti

23.3.2018

Tekijä Otsikko	Jarkko Aho OBD-etälukulaite
Sivumäärä Aika	10 sivua + 3 liitettä 23.3.2018
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintäteknikka
Ammatillinen pääaine	Smart Systems
Ohjaajat	Lehtori Kimmo Sauren Projektimanageri Nikita Pavlov
<p>Innovaatioprojektina tehtiin asiakastutkimustyö startup-yritykselle, joka keskittyy ajoneuvojen itsediagnostiikkalaitteisiin eli OBD-laitteisiin (On-Board Diagnostics). Työn tavoitteena oli suunnitella auton OBD-väylään kiinnitettävä laite, joka keräisi haluttua tietoa ajoneuvon toiminnasta ja lähettäisi sen puhelinverkon kautta yrityksen palvelimelle.</p> <p>Projektia lähdettiin suunnittelemaan asiakkaan asettamien vaatimusten mukaisesti. Työn osat valittiin niiden saatavuuden ja helppokäyttöisyyden perusteella. Valmistuneen prototyypin tarkoituksena oli selvittää myöhempään tuotekehitykseen liittyvät ongelmat ja tuotteen valmistamisen laajuus.</p> <p>Ohjelmisto suunniteltiin C/C++-ohjelmointi kielellä LPCXpresso 1549 -mikroprosessorille ja ohjelmoitiin MCUXpresso IDE -ohjelmointiympäristössä. Ajoneuvon kiihtyvyydestä keräsi erillinen kiihtyvyyssanturi, sijaintitietoa erillinen GPS-moduuli ja tiedot lähetettiin 3G-puhelinverkkoon erillisellä GPRS/GSM-moduulilla.</p> <p>Lopputulosta ei ehditty liittää ajoneuvon OBD-väylään, eikä tiedon lähettämistä yrityksen palvelimelle ehditty testaamaan projektin laajuuden rajoissa. Valmistunut prototyyppi keräsi kiihtyvyy- ja sijaintitietoja ja pystyi ottamaan 3G-yhteyden internetiin.</p>	
Avainsanat	OBD, GPS, GPRS, IoT, auto, ajoneuvo, LPC

Author Title	Jarkko Aho Remote reading OBD-device
Number of Pages Date	10 pages + 3 appendices 23 March 2018
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Smart Systems
Instructors	Kimmo Sauren, Senior Lecturer Nikita Pavlov, Project Manager
<p>This innovation project was a customer research project for startup company that specializes in OBD-devices (On-Board Diagnostics). The aim of this project was to design a OBD-connected device that collects data from the vehicle and sends it to company's server using cellular network.</p> <p>The project design was started according to the customer's requirements. Parts of the project were selected based on their availability and ease of use. The purpose of the completed prototype was to find out the future problems of product development and the extent of product manufacturing.</p> <p>The wireframe was designed with C/C++ programming language to LPCXpresso 1549 - microprocessor and was programmed using the MCUXpresso IDE programming environment. Vehicle acceleration and location data was recorded with a separate acceleration and GPS modules and data was send to 3G cellular network with a separate GPRS/GSM module.</p> <p>The result was not tested on OBD-connection nor the data was sent to company's server, due to an innovation project's extent. The resulted prototype collected the acceleration and location data and could connect to the internet with 3G.</p>	
Keywords	OBD, GPS, GPRS, IoT, auto, vehicle, LPC

# Sisällys

## Lyhenteet

1	Johdanto	1
2	Prototyypissä käytetyt komponentit	1
2.1	NXP LPCXpresso LPC1549 -kehitysalusta	1
2.2	Maestro A2035-H -GPS-moduuli	2
2.3	Sparkfun MPU9250 -liiketunnistinmoduuli	2
2.4	Seeedstudio GPRS/GSM Shield V2.0 -kilpi	3
2.5	Komponentit valmiissa tuotteessa	4
3	Prototyypissä käytetyt ohjelmistot	4
3.1	MCUXpresso IDE -ohjelmointiympäristö	4
3.2	FreeRTOS Kernel -käyttöjärjestelmäydin	5
4	Koodin toimintaperiaate	5
4.1	Kiihtyvyysanturi-tehtävä	6
4.2	GPS-tehtävä	7
4.3	GPRS-tehtävä	8
5	Prototyyppi ja laajentamismahdollisuudet	8
6	Yhteenveto	9
	Lähteet	10

## Liitteet

Liite 1. AT-komentotesti

Liite 2. KytKentäkaavio

Liite 3. Lähdekoodi

## Lyhenteet

3G	Third Generation. Kolmannen sukupolven langaton matkaviestinteknologia.
ARM	Advanced RISC Machine. Mikroprosessoriarkkitehtuuri.
AT	Attention command (AT-command). GSM/GPRS-modeemien ohjauskäskysarja.
GPIO	General-Purpose Input Output. Yleiskäyttöinen vastaanotin-lähetinportti mikroprosessoreissa.
GPRS	General Packet Radio Service. GSM-verkossa toimiva pakettikytkentäinen tiedonsiirtopalvelu.
GPS	Global Positioning System. Maailmanlaajuinen paikallistamisjärjestelmä.
GSM	Global System for Mobile Communications. Maailmanlaajuinen matkapuhelinjärjestelmä.
I <sup>2</sup> C	Inter-Integrated Circuit. Kaksisuuntainen ohjaus- ja tiedonsiirtoväylä mikroprosessorin ja komponentin välillä.
IDE	Integrated Development Environment. Ohjelmointiympäristö, jolla ohjelmoija suunnittelee ja toteuttaa ohjelmistoja.
IoT	Internet of Things. Esineiden internetillä tarkoitetaan internetverkon laajentumista laitteisiin ja koneisiin.
ISO	International Organization for Standardization. Kansainvälinen standardisoimisjärjestö.
LPC1549	NXP LPCXpresso LPC1549. Työssä käytetty mikroprosessori.
MCU	Motion Control Microcontroller. Mikrokontrolleriyksikkö eli mikroprosessori.

NMEA	The National Marine Electronics Associationin mukaan nimetty GPS-tietopaketti, jota GPS-moduuli lähettää mikroprosessorille.
OBD	On-Board Diagnostics. Ajoneuvojen itsediagnostiikkajärjestelmä.
RTOS	Real-Time Operating System. Reaaliaikainen käyttöjärjestelmä.
Rx	Receiver. Vastaanottava liitäntä UART-väylässä.
SCL	Clock-liitäntä I <sup>2</sup> C-väylässä.
SDA	Dataliitäntä I <sup>2</sup> C-väylässä.
SIM	Subscriber Identity Module. SIM-kortti, joka löytyy jokaisesta kuluttajakäytössä olevasta matkapuhelimesta.
SPI	Serial Peripheral Interface bus. Synkronoitu sarjaliikenneväylä.
Tx	Transmitter. Lähettävä liitäntä UART-väylässä.
UART	Universal Asynchronous Receiver-Transmitter. Sarjaliikenneväylä mikroprosessorin ja komponentin välillä.
USB	Universal Serial Bus. Sarjaväyläarkkitehtuuri oheislaitteiden liittämiseksi tietokoneeseen.

## 1 Johdanto

IoT (Internet of Things, esineiden internet) on kehittynyt huomattavasti viime vuosien aikana jokaisella tekniikan ja tuotannon alueella, erityisesti ajoneuvotekniikassa. Jokainen nykyaikainen ajoneuvo sisältää jopa satoja erilaisia oheislaitteita, jotka voivat olla kytköksissä internetiin. Suurinta osaa ajoneuvon oheislaitteista voidaan lukea ajoneuvon OBD-väylästä (On-Board Diagnostics), joka on standardoitu ISO-standardilla.

Servoped Oy haluaa kilpailla nopeasti kasvavia yhteiskäyttöajoneuvomarkkinoita vastaan ja tuoda asiakkailleen mahdollisuuden saada tietoa ajoneuvon toiminnasta reaaliajassa. Vaikka suurin osa ajoneuvon oheislaitteiden tiedosta on luettavissa OBD-väylästä, tietoihin käsiksi pääseminen ei ole helppoa tavalliselle autoilijalle.

Innovaatioprojektin tarkoituksena oli suunnitella ja toteuttaa OBD-väylään kiinnitettävä etälukulaite, joka keräisi ajoneuvon OBD-tietojen lisäksi ajoneuvon kiihtyvyystietoja ja GPS-sijainnin (Global Positioning System) ja lähettäisi kaiken yrityksen pilvipalveluun GSM-verkon (Global System for Mobile Communications) kautta.

## 2 Prototyypissä käytetyt komponentit

### 2.1 NXP LPCXpresso LPC1549 -kehitysalusta

LPC1549 on NXP Semiconductorsin ARM Cortex-M3 -pohjainen kehitysalusta, joka on suunniteltu erityisesti sulautetuille järjestelmille. Kehitysalustan prosessori toimii säädettävällä 72 MHz:n taajuudella ja sisältää 256 kB flash-muistia [1, s. 1]. Nämä ominaisuudet mahdollistavat sen yhteensopivuuden lähes kaikkien ulkopuolisten sensorien kanssa.

LPC1549:ssä on kolme säädettävää UART-liitäntää (Universal Asynchronous Receiver-Transmitter) ja yksi säädettävä I<sup>2</sup>C-liitäntä (Inter-Integrated Circuit), joita kaikkia käytettiin tässä työssä [1, s. 2]. Kaksi UART-liitäntää oli kytketty GPS- ja GSM moduuleille, yksi UART-liitäntä toimi virhetestauksen debug-porttina ja I<sup>2</sup>C-liitäntä oli kytketty kiihtyvyyssanturille.

LPC1549-kehitysalusta valittiin täysin sen muokattavuuden ja saatavuuden takia, ja se mahdollisti kaikkien ulkopuolisten moduulien helpon asentamisen työn aikana. LPC1549 on hyvä vaihtoehto valmiin tuotteen tuotantoon, mutta myös pienempi prosessori riittäisi. Esimerkiksi NXP Semiconductors ja Cypress valmistavat monta pienempää mikroprosessoria, joilla on vastaavat liitäntäominaisuudet kuin työssä käytetyllä LPC1549:llä.

## 2.2 Maestro A2035-H -GPS-moduuli

Maestron A2035-H-GPS-moduuli on hyvin yksinkertainen GPS-paikannuslaite. Moduulissa on sisäänrakennettu prosessori, joka heti käynnistyessään etsii automaattisesti GPS-satelliitit ja antaa NMEA-sijaintitietoa (National Marine Electronics Association) UART-sarjaliikenneliitännän kautta. Moduulissa oli myös mahdollisuus SPI-liitännälle (Serial Peripheral Interface bus).

Moduulin suurin ongelma on sen muistin korruptoitumisen vaara. Jos moduulin virta katkeaa yllättäen, sen muisti saattaa korruptoitua ja laitteesta tulee käyttökelvoton. Tämän takia moduulin reset-pinniin asennettiin Maxim MAX809SEUR+T -virranvalvontapiiri, joka pitää huolen moduulin sammuttamisesta turvallisesti yllättävän virtakatkon aikana [2, s. 18; 3].

A2035-H on erityisen hyvä vaihtoehto valmiille tuotteelle sen hyvin pienen koon ja helppokäyttöisyytensä ansiosta.

Työtä testattiin myös Fastrax uPatch100-S -GPS-moduulin kanssa, jonka toimintaperiaate on sama kuin Maestron moduulissa. Fastrax ei tarvitse virranvalvontapiiriä, mutta on huomattavasti isompi kuin Maestro.

## 2.3 Sparkfun MPU9250 -liiketunnistinmoduuli

MPU9250-moduuli on InvenSensen valmistama vähävirtainen liiketunnistin, joka sisältää kolmiakselisen kiihtyvyysanturin, -gyroskoopin ja -magnetometrin. MPU9250:stä käytettiin vain kolmiakselista kiihtyvyysanturia, jota ohjaa MPU9250-piirin sisäinen MPU6500-

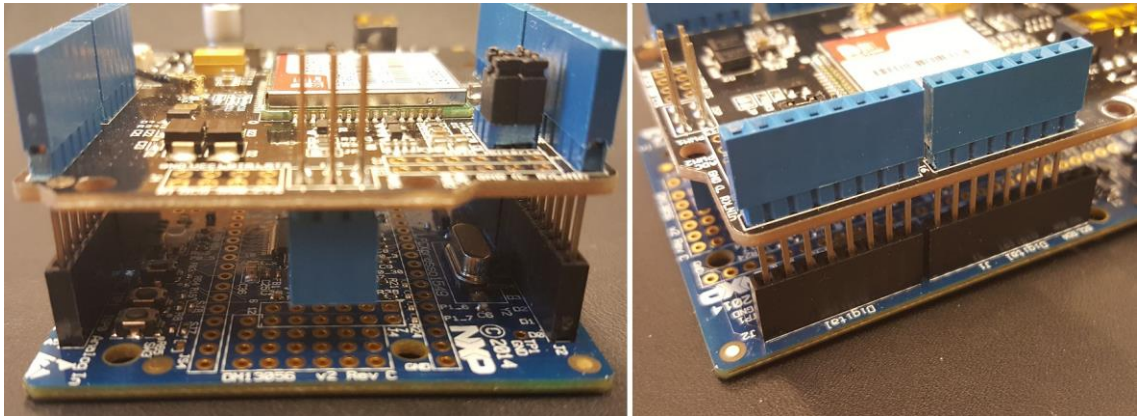


prosessori [4, s. 6; 5]. MPU9250 liitettiin LPC1549:ään I<sup>2</sup>C-liitännällä, joka eroaa UART-liitännästä sen oman kellotaajuudenohjauspinnin osalta. Moduulissa oli myös mahdollisuus SPI-liitännälle.

Kiihtyvyysanturi ei ollut täysin automaattinen kuten edeltävät moduulit. Kiihtyvyysanturi vaati käynnistyskäskyt ja kalibroinnin tarkan arvon saamiseksi. Näiden käskyjen antaminen tuotti eniten ongelmia moduulin käyttöönotossa. Lähes kaikki moduulin toiminta-esimerkit oli tehty Arduino-mikroprosessoria käyttäen, jonka C#-ohjelmointikielen esimerkkikoodit eivät vastanneet LPC1549:ssä käytettävää C/C++-ohjelmointikieltä tai ohjelmointiympäristön kirjastoja. Esimerkkikoodin funktiot kirjoitettiin uudelleen C/C++-ohjelmointikielellä, ja kiihtyvyysanturi saatiin toimimaan halutusti.

## 2.4 Seeedstudio GPRS/GSM Shield V2.0 -kilpi

Tietojen lähettämisestä palvelimelle vastasi Seeedstudion Arduinolle tarkoitettu GPRS-kilpi (General Packet Radio Service). GPRS-kilvessä oli SIM-kortti, jonka avulla laite oli tarkoitus yhdistää 3G GPRS -verkkoon ja Servoped Oy:n palvelimelle. GPRS-kilpi liitettiin LPC1549:ään UART-liitännällä. LPC1549:ssä on sama pinnijärjestys kuin Arduinossa (kuva 1), joten GPRS-kilpi ei tarvinnut erillisiä hyppylankoja.



Kuva 1. LPC1549:ssä (kuvissa alempi piirilauta) on sama pinnijärjestys kuin Arduinolle suunnitellussa GPRS-kilvessä (kuvissa ylempi piirilauta) [6].

GPRS-kilvessä olevaa SIM900 GPRS/GSM -moduulia ohjataan AT-komennoilla (Attention command) [6; 7]. SIM900-moduulin AT-komennot eroavat hieman tyypillisistä AT-komennoista, ja niiden oppiminen vei paljon aikaa. AT-komentojen lähettäminen UART-

liitännällä oli helppoa, mutta SIM900-moduulin toimintaa oli todella vaikea kokeilla ilman vapaassa käytössä olevaa palvelinta. Ilman toimivaa palvelinta työssä tehtiin vain yksinkertainen yhteyspyyntö Googlen palvelimelle (liite 1).

Työssä käytetty GPRS-kilpi oli Seeedstudioin vanhempi malli [8]. Molemmissa versioissa on sama SIM900 GPRS/GSM -moduuli.

## 2.5 Komponentit valmiissa tuotteessa

Työtä ei ehditty kytkemään OBD-moduuliin, joka lukisi tietoa OBD-väylästä. Tämän moduulin asentaminen vaatisi viimeisen UART-liitännän LPC1549:stä, jolloin debug-toiminto ei olisi mahdollinen. Debug-toimintoa ei kuitenkaan tarvita, jos tuote on GSM-yhteydessä pilvipalveluun. Tällöin laitteessa syntyvät virheet voitaisiin lähettää suoraan pilvipalveluun, josta Servoped Oy tai asiakas saisi mahdollisen virheilmoituksen viallisesta tuotteesta.

Työstä syntynyt prototyyppi sai virtansa USB-johdolla (Universal Serial Bus), josta lähtevä 5 voltin jännite jaettiin ulkopuolisille moduuleille LPC1549:n kautta. Valmis tuote on tarkoitus kiinnittää OBD-väylään, josta lähtevä 12 voltin jännite rikkoisi sekä LPC1549:n että kaikki moduulit. Siksi valmis tuote vaatii jännitteenjakajan, joka jakaisi 12 voltin jännitteen 5 voltin jännitteeseen.

## 3 Prototyypissä käytetyt ohjelmistot

### 3.1 MCUXpresso IDE -ohjelmointiympäristö

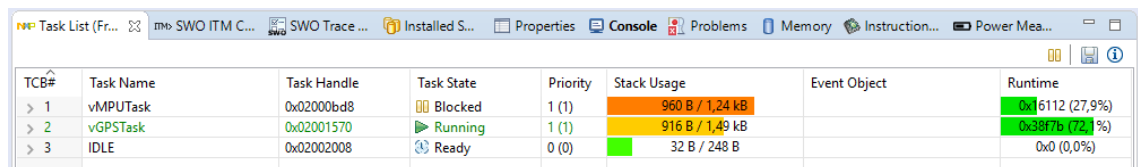
MCUXpresso IDE -ohjelmointiympäristö on NXP Semiconductorsin Eclipse-pohjainen kehitysympäristö, joka on suunniteltu erityisesti NXP Semiconductorsin ARM Cortex-M -kehitysalustoille. MCUXpresso-ohjelmointiympäristö tarjoaa monipuoliset muokkaus-, kokoamis- ja virheidenkorjausominaisuudet sekä monipuoliset työkalut koodinseurantaan [9, s. 8]. MCUXpresso IDE valittiin ohjelmointiympäristöksi, koska se tukee LPC1549-kehitysalustaa, se on ilmainen pienissä projekteissa ja mahdollistaa helpot virheidenkorjaustarkistukset.

### 3.2 FreeRTOS Kernel -käyttöjärjestelmäydin

FreeRTOS Kernel (Real-Time Operating System) on Real Time Engineersin kehittämä reaaliaikainen käyttöjärjestelmäydin (kernel). FreeRTOS:llä on mahdollista jakaa koodi tehtäviin (task), joista jokainen tehtävä vastaa yhdestä moduulista [10, s. 32–36]. FreeRTOS alustaa jokaisen tehtävän automaattisesti ja ajastaa ne toimimaan ohjelmoidussa järjestyksessä.

FreeRTOS:n opasteilla (semaphore) pidettiin huolta, että useampi tehtävä ei kirjoita debug UART -väylään samaan aikaan [10, s. 210–212]. Näin varmistettiin, että tallennettu tieto ei korruptoidu.

FreeRTOS mahdollisti myös tehtäväkohtaisen virheenkorjauksen. FreeRTOS tarjosi oman debug-ympäristönsä (Task List), jolla kyettiin seuraamaan jokaista tehtävää yksilöllisesti ja paikantamaan mahdolliset virheet tehtäväkohtaisesti (kuva 2).

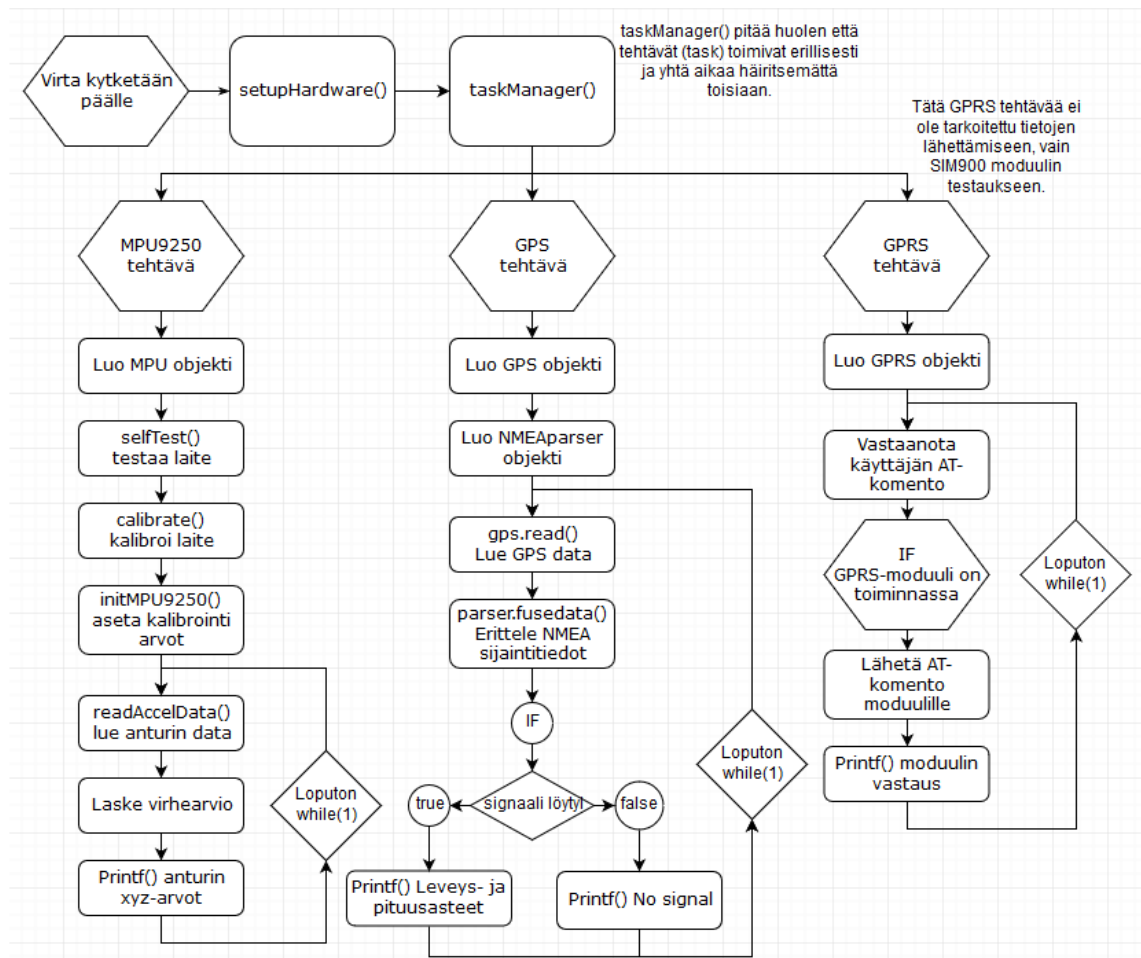


TCB#	Task Name	Task Handle	Task State	Priority	Stack Usage	Event Object	Runtime
> 1	vMPUTask	0x02000bd8	Blocked	1 (1)	960 B / 1,24 kB		0x00000000 (0,0%)
> 2	vGPSTask	0x02001570	Running	1 (1)	916 B / 1,49 kB		0x39f7b (72,1%)
> 3	IDLE	0x02002008	Ready	0 (0)	32 B / 248 B		0x0 (0,0%)

Kuva 2. FreeRTOS Task List, jolla seurataan jokaista tehtävää (task) yksilöllisesti [10, s. 94–96].

## 4 Koodin toimintaperiaate

Työ ohjelmointiin C/C++-ohjelmointikielellä, ja ohjelmakoodi toimii RTOS-periaatteella. Työ ohjelmointiin toimimaan heti, kun virta kytketään päälle. Ohjelma alustaa itsensä ja kaikki ulkopuoliset laitteet automaattisesti. FreeRTOS:n tehtävä (task) -ominaisuudella ohjelma jaettiin tehtäviin, joista jokainen tehtävä vastasi erillisestä moduulista. Kuvassa 3 on ohjelmakoodin vuokaavio, josta näkee koodin toimintaperiaatteen. Vuokaavio esittää ohjelman pääfunktion (main) toimintaa. Ohjelmakoodin pääfunktio tiedosto kokonaisuudessaan on liitteessä 3.



Kuva 3. Ohjelmakoodin vuokaavio.

Pääfunktio alustaa ensin kehitysalustan asetukset, minkä jälkeen FreeRTOS:n vTaskStartScheduler-funktio käynnistää jokaisen tehtävä-funktion (task). Jokainen tehtävä-funktio ohjaa erillistä moduulia luomalla ensin moduulille tarkoitetun objektin.

#### 4.1 Kiihtyvyysanturi-tehtävä

MPU9250-moduulin objekti-luokka muutettiin Arduinolle tehdystä koodista, joka käyttää C#-ohjelmointikieltä [11]. Kun MPU-objekti luodaan, luokaa alustavat LPC1549-kehitysalustan pinnit 0.22 ja 0.23 I<sup>2</sup>C-tilaan, joista pinni 0.22 toimii moduulin synkronointikellona (SCL) ja pinni 0.23 tiedonsiirtoväylänä (SDA).

Kiihtyvyyssanturi-tehtävä valmistelee moduulin seuraavasti:

- selfTest-funktio vertaa moduulin kalibrointiasetuksia tehdasasetuksiin ja tallentaa asetusten eron virhearviota varten.
- calibrate-funktio kalibroi moduulin ottaen samalla huomioon Z-akselin suuntaisen maan vetovoiman.
- initMPU9250-funktio tallentaa kalibroidut arvot moduuliin ja valmistelee laitteen käyttöönottamista varten.

Tämän jälkeen tehtävä aloittaa loputtoman while-silmukan, jonka sisällä tehtävä lukee kiihtyvyyssanturin X-, Y- ja Z-arvot, laskee niiden todellisen arvon virhearviota käyttäen ja lopuksi tulostaa arvot käyttäjälle debug-väylään. Tämä silmukka toistuu loputtomasti, kunnes laitteesta katkeaa virta.

#### 4.2 GPS-tehtävä

GPS-luokaa alustavat LPC1549-kehitysalustan pinnit 0.29 (Rx) ja 0.9 (Tx) UART-tilaan. UART-liitäntä ei tarvitse synkronointikelloliitäntää kuten I<sup>2</sup>C, mutta vaatii, että molemmat laitteet on kytketty samaan nollan voltin maajohtoon (ground). GPS-luokka alustaa samalla NMEAParser-objektin, joka erittelee GPS-moduulin lähettämät tiedot.

GPS-tehtävän loputon while-silmukka lukee NMEA-tiedot moduulilta ja tarkistaa, onko moduuli yhteydessä satelliitteihin. Jos moduuli on löytänyt satelliitit, NMEAParser-objekti erittelee NMEA-tiedoista laitteen sijainnin pituus- ja leveysasteet, jotka tulostetaan käyttäjälle debug-väylään [12; 13 s. 11–15]. Jos satelliitteja ei löytynyt, tulostetaan teksti "No signal". Tämä silmukka toistetaan, kunnes laitteesta katkeaa virta.

Löytääkseen satelliitit moduulin on oltava lähes esteettömässä yhteydessä taivaalle noin viisitoista minuutin ajan virran päälle kytkemisen jälkeen.

#### 4.3 GPRS-tehtävä

Työssä käytetty GPRS-tehtävä luo GPRS-objektin ja alustaa sen UART-tilan pinneihin 1.10 (Rx) ja 1.9 (Tx). GPRS-moduulia ei ehditty kokeilemaan sille tarkoitettussa käytössä. GPRS-tehtävän while-silmukka odottaa käyttäjän AT-komentoa ja lähettää sen GPRS-moduulille, jos moduuli on toiminnassa. GPRS-moduulin vastaus vastaanotetaan ja tulostetaan käyttäjälle debug-väylään. Tämä tehtävä ei toimi automaattisesti, kuten kiihtyvyysanturi- ja GPS-tehtävät, eikä se sovellu tuotteelle tarkoitettuun käyttöön. GPRS-tehtävää ei voi käyttää samaan aikaan kiihtyvyysanturi- ja GPS-tehtävän kanssa, koska GPRS-tehtävä vaatii käyttäjältä AT-komennon syöttämistä debug-väylään.

GPRS-tehtävän tarkoitus olisi alustaa GPRS-moduuli lähettämällä tarvittavat AT-komennot Servoped Oy:n palvelimeen yhdistämiseksi ja jatkaa loputtomaan while-silmukkaan, jossa tehtävä odottaisi kiihtyvyysanturi-, GPS- ja OBD-tietoja. Tiedot saatuaan tehtävä paketoisi tiedot yhdeksi merkkijonoksi (string) ja lähettäisi sen valmiiksi ohjelmoidulla AT-komennolla Servoped Oy:n palvelimelle.

### 5 Prototyyppi ja laajentamismahdollisuudet

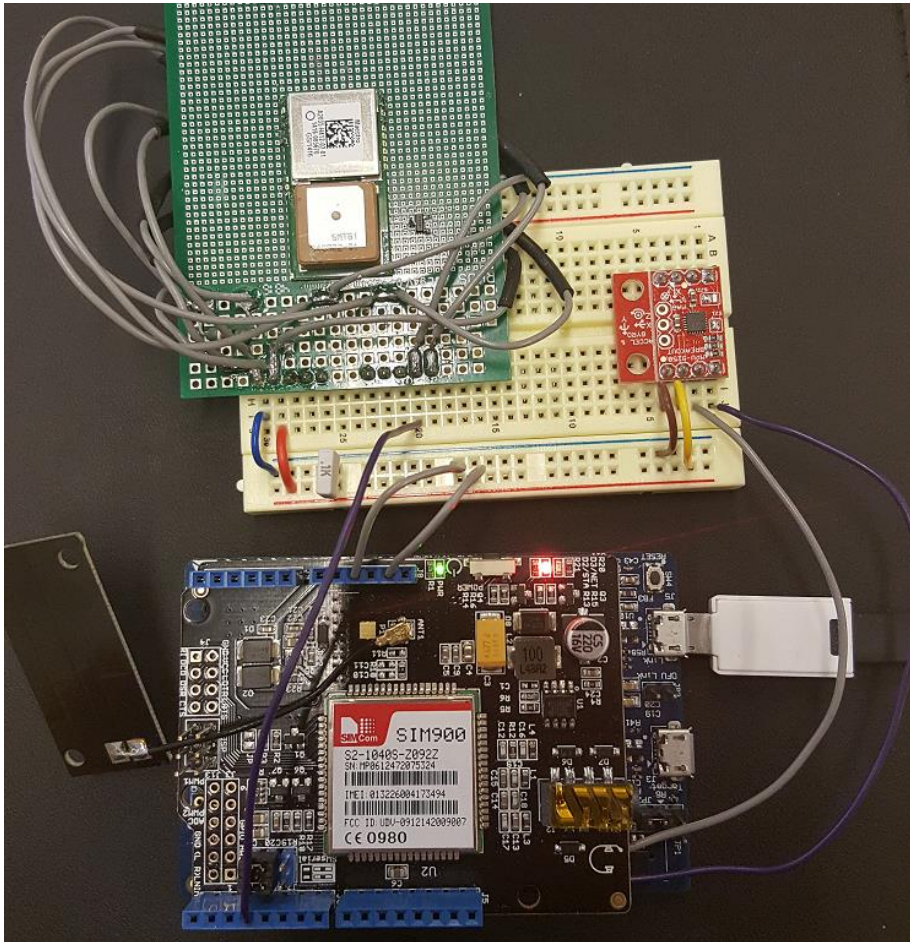
Työssä valmistuneessa prototyypissä olevat moduulit toimivat kuten suunniteltiin. GPS-moduuli ja sen jäsenninfunktio toimivat automaattisesti järjestelmän käynnistyessä. Ilman tarkempaa antennia GPS-moduuli tarvitsi esteettömän näkyvyyden taivaalle satelliittien löytämiseksi. Kiihtyvyysanturi kalibroitiin ja antoi tarkkoja arvoja. Kiihtyvyysanturin kalibrointi on tehtävä uudelleen, jos laite sijoitetaan paikkaan, jossa x- ja y-akseli eivät ole vaakasuorassa kulmassa maanvetovoimaan nähden.

Laitteen GPRS-moduulia ei pystytty testaamaan ilman toimivaa palvelinta. GPRS-moduuli saatiin silti toimimaan, ja sillä saatiin yhteyspyyntö Googlen-palvelimeen. Työ vaatii jatkokehitystä GPRS-moduulin ja AT-komentojen toimintaan. Työssä ei tehty toimivaa tehtävfunktiota GPRS-moduulin käyttämiseen.

Työtä ei testattu sen alkuperäisessä tarkoituksessa OBD-etälukulaitteena. Laitteeseen oli tarkoitus kiinnittää vielä OBD-moduuli, jolla olisi saatu ajoneuvon OBD-tietoja, mutta haluttua moduulia ei ollut saatavilla. Jatkokehityksessä OBD-moduulille on tehtävä oma tehtävfunktio ja alustettava omat UART-liitännät.

## 6 Yhteenveto

Projektin osat toimivat kuten piti, ja syntynyt prototyyppi antoi karkean suunnitelman valmiin OBD-etälukulaitteen tuotannosta, mutta luokittelen projektin epäonnistuneeksi. Tuote vaatii paljon jatkokehitystä, ja tuotteeseen valittaville osille on suunniteltava piirilauta, koska työstä syntynyt prototyyppi ei sovellu edes testikäyttöön (kuva 4).



Kuva 4. OBD-etälukulaitteen prototyyppi. Ylhäältä alas: GPS-moduuli, kiihtyvyyssanturi, GPRS-kilpi, LPC1549-kehitysalusta.

Projekti ei onnistunut tarvittavien osien ja pilvipalvelimen puutteen takia. Projekti oli hyvin haastava, koska työ tehtiin innovaatioprojektina, jolle varattu aika oli hyvin rajallinen. Projekti sisälsi eniten ohjelmointiosaamista, mutta myös laajempi elektroniikka- ja ajoneuvotietämys olisi ollut projektissa eduksi. Servoped Oy sai työstä hyvän toimintasuunnitelman tuotteen valmistamisen ja jatkokehityksen laajuudesta.



## Lähteet

- 1 LPC15xx Product datasheet. 2015. LPC15xx Product datasheet revision 1.1. E-kirja. NXP Semiconductors.
- 2 Happy, Wen. 2013. GPS Receiver A2035-H User's Manual version 1.4. E-kirja. Maestro.
- 3 MAX803-MAX810Z datasheet. 2017. MAX803/MAX809/MAX810 datasheet revision 11. E-kirja. Maxim integrated.
- 4 MPU-9250 Product Specification. 2014. MPU-9250 Product Specification Revision 1.0. E-kirja. InvenSense.
- 5 MPU-9250 Hookup Guide. 2014. Verkkoaineisto. SparkFun. <[https://learn.sparkfun.com/tutorials/mpu-9250-hookup-guide?\\_ga=2.31321084.2140885038.1516008089-1929693235.1482834715](https://learn.sparkfun.com/tutorials/mpu-9250-hookup-guide?_ga=2.31321084.2140885038.1516008089-1929693235.1482834715)>. Luettu 22.3.2018.
- 6 Devrhoid, Davis. 2015. AT Command set for sending data via TCP using SIM900. Verkkoaineisto. SlideShare. <<https://www.slideshare.net/DevrhoidDavis1/at-command-set-for-sending-data-via-tcp-using-sim900>>. Päivitetty 18.5.2015. Luettu 22.3.2018.
- 7 SIM900 AT Command Manual. 2010. SIM900 AT Command Manual version 1.03. E-kirja. SIMCom Wireless Solutions.
- 8 GPRS Shield V2.0. 2010. Verkkoaineisto. Seeedstudio. <[https://seeeddoc.github.io/GPRS\\_Shield\\_V2.0/](https://seeeddoc.github.io/GPRS_Shield_V2.0/)>. Luettu 22.3.2018.
- 9 MCUXpresso IDE User Guide. 2017. MCUXpresso IDE User Guide revision 10.1.0. E-kirja. NXP Semiconductors.
- 10 FreeRTOS V9.0.0 Reference Manual. 2016. Reference Manual for FreeRTOS version 9.0.0 issue 2. E-kirja. Real Time Engineers.
- 11 Wilkins, Brent. 2016. SparkFun MPU-9250 Breakout Arduino Library. Verkkoaineisto. Github. <[https://github.com/sparkfun/SparkFun\\_MPU-9250\\_Breakout\\_Arduino\\_Library](https://github.com/sparkfun/SparkFun_MPU-9250_Breakout_Arduino_Library)>. Päivitetty 7.2.2017. Luettu 22.3.2018.
- 12 Radu, Motisan. 2013. AVR NMEA GPS library. Verkkoaineisto. Github. <<https://github.com/radhoo/avr-nmea-gps-library>>. Päivitetty 8.5.2017. Luettu 22.3.2018.
- 13 NMEA Reference Manual. 2005. NMEA Reference Manual Revision 1.3. E-kirja. SiRF Technology.



## AT-komentotesti

Tässä on esimerkki työssä käytetyistä AT-komennoista, joilla tehtiin yksinkertainen yhteyspyyntö Googlen palvelimeen. Jokaisen AT-komennon edessä täytyi olla "AT", jonka jälkeen lisättiin itse komento, esimerkiksi "+CREG?". Ilman komentoa moduuli vastasi vain "OK", jos teksti "AT" otettiin vastaan onnistuneesti. Liitteen toisella sivulla selitetään käytettyjen komentojen toimintatarkoitukset.

```
AT
OK
AT+CREG?
+CREG: 1,1
OK
AT+CGATT?
+CGATT: 1
OK
AT+CIPSTATUS
OK
STATE: IP INITIAL
AT+CIPMUX=0
OK
AT+CIPSHUT
SHUT OK
AT+CIPSTART="TCP", "WWW.GOOGLE.COM", "80"
OK
CONNECT OK
AT+CIPSHUT
SHUT OK
```

Komento: AT  
Vastaus: OK  
Selitys: Kokeilee kahden laitteen välistä kommunikaatiota.

Komento: AT+CREG?  
Vastaus: +CREG=1,1  
OK  
Selitys: SIM-kortti on valmis ja yhdistetty GPRS-verkkoon.

Komento: AT+CGATT?  
Vastaus: +CGATT: 1  
Selitys: SIM-kortilla on yhteys internetiin.

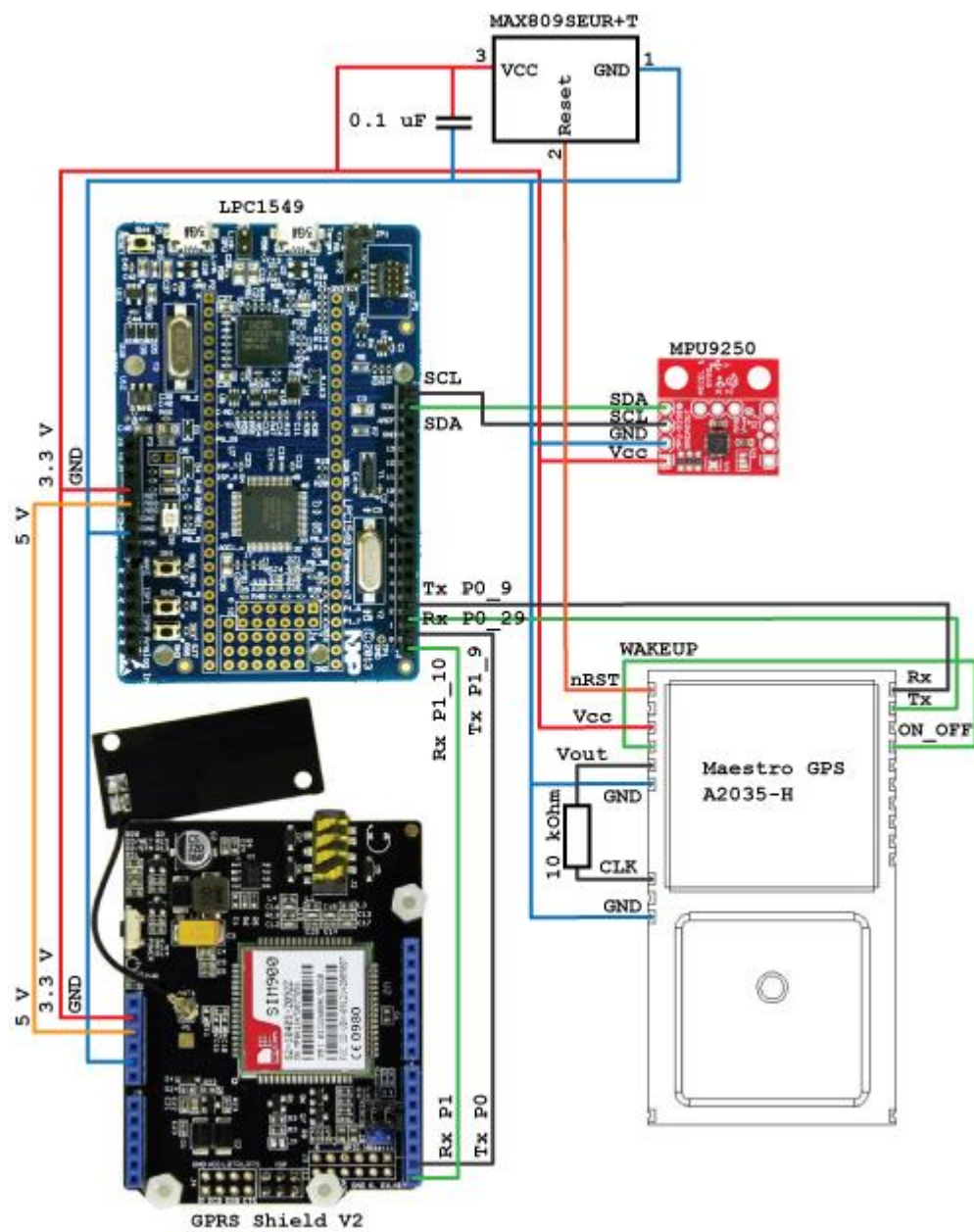
Komento: AT+CIPSTATUS?  
Vastaus: OK  
STATE: IP INITIAL  
Selitys: IP-asetusten tiedustelu.

Komento: AT+CIPMUX=0  
Vastaus: OK  
Selitys: Asettaa modeemin yksittäisen portin avoimeen yhteyteen.

Komento: AT+CIPSHUT  
Vastaus: SHUT OK  
Selitys: Sulkee TCP-portin nimenomaisesti.

Komento: AT+CIPSTART="TCP","WWW.GOOGLE.COM","80"  
Vastaus: OK  
CONNECT OK  
Selitys: Tekee TCP-yhteyden annettuun palvelimeen portista 80.

## Kytentäkaavio



## Lähdekoodi

```

/** MAIN FILE */
#if defined (__USE_LPCOPEN)
#if defined(NO_BOARD_LIB)
#include "chip.h"
#else
#include "board.h"
#endif
#endif

#include <cr_section_macros.h>
#include <cctype>          // toupper() function for AT commands

#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"

#include "MPU9250.h"      // pins assigned to SCL 0.22 - SDA 0.23
#include "NMEA.h"         // pins assigned to rx 0.9 - tx 0.29
#include "NMEAparser.h"   // getter functions for parsed NMEA data
#include "GPRS.h"         // pins assigned to rx 1.10 - tx 1.9

#define DEVICE_NUMBER    0          // MPU9250 I2C device number
#define SPEED             100000     // MPU9250 I2CM transfer rate

// Global semaphore for printf() functions
SemaphoreHandle_t xBin;

// LPC board setup
static void prvSetupHardware(void) {
    SystemCoreClockUpdate();
    Board_Init();
    Board_LED_Set(0, false);
    Chip_RIT_Init(LPC_RITIMER);
}

extern "C" {
void vConfigureTimerForRunTimeStats(void) {
    Chip_SCT_Init(LPC_SCTSMALL1);
    LPC_SCTSMALL1->CONFIG = SCT_CONFIG_32BIT_COUNTER;
    LPC_SCTSMALL1->CTRL_U = SCT_CTRL_PRE_L(255) | SCT_CTRL_CLRCTR_L;
}
}

// Accelerometer task
static void vMPUTask(void *pvParameters) {
    // Create MPU9250 object
    MPU9250 myMPU(DEVICE_NUMBER, SPEED);

    // Init self test and print values for debugging
    myMPU.selfTest(myMPU.selfTestArray);
    xSemaphoreTake(xBin, 10);
    printf("Self Test:\r\n");
    printf("x-axis self test: acceleration trim within : ");
    printf("%f", myMPU.selfTestArray[0]); printf("%% of factory value\r\n");
    printf("y-axis self test: acceleration trim within : ");
    printf("%f", myMPU.selfTestArray[1]); printf("%% of factory value\r\n");
    printf("z-axis self test: acceleration trim within : ");
    printf("%f", myMPU.selfTestArray[2]); printf("%% of factory value\r\n");
    printf("x-axis self test: gyration trim within : ");
    printf("%f", myMPU.selfTestArray[3]); printf("%% of factory value\r\n");
    printf("y-axis self test: gyration trim within : ");

```

```

printf("%f", myMPU.selfTestArray[4]); printf("%% of factory value\r\n");
printf("z-axis self test: gyration trim within : ");
printf("%f", myMPU.selfTestArray[5]); printf("%% of factory value\r\n");
xSemaphoreGive(xBin);

// Calibrate device and print bias values
myMPU.calibrate(myMPU.gyroBias, myMPU.accelBias);
xSemaphoreTake(xBin, 10);
printf("MPU9250 Bias:\r\n");
printf("x: %f mg, y: %f mg, z: %f mg\r\n", (1000*myMPU.accelBias[0]),
(1000*myMPU.accelBias[1]), (1000*myMPU.accelBias[2]));
printf("x: %f o/s, y: %f o/s, z: %f o/s\r\n", myMPU.gyroBias[0],
myMPU.gyroBias[1], myMPU.gyroBias[2]);
printf("\r\n");
xSemaphoreGive(xBin);

// Init device based on calibration values
myMPU.initMPU9250();

// Init magnetometer for debugging
myMPU.initAK8963(myMPU.factoryMagCalibration);
xSemaphoreTake(xBin, 10);
printf("X-Axis sensitivity adjustment value: %f\r\n", myMPU.factoryMa-
gCalibration[0]);
printf("Y-Axis sensitivity adjustment value: %f\r\n", myMPU.factoryMa-
gCalibration[1]);
printf("Z-Axis sensitivity adjustment value: %f\r\n", myMPU.factoryMa-
gCalibration[2]);
printf("\r\n");
xSemaphoreGive(xBin);

// Task main loop
// Prints the accelerator values
// These values can be send to GPRS
while(1) {
    myMPU.readAccelData(myMPU.accelCount);
    myMPU.getAres();

    myMPU.ax = (float)myMPU.accelCount[0]*myMPU.aRes;
    myMPU.ay = (float)myMPU.accelCount[1]*myMPU.aRes;
    myMPU.az = (float)myMPU.accelCount[2]*myMPU.aRes;

    xSemaphoreTake(xBin, 10);
    printf("X: %f mg, Y: %f mg, Z: %f mg\r\n", (1000*myMPU.ax),
(1000*myMPU.ay), (1000*myMPU.az));
    xSemaphoreGive(xBin);

    vTaskDelay(configTICK_RATE_HZ / 2);
}

// GPS NMEA task
static void vGPSTask(void *pvParameters) {
    NMEA gps;
    NMEAParser parser;
    bool end;
    while(1) {
        end = false;
        while (end == false) {
            int r = gps.read();
            if (r != -1) {
                parser.fusedata(r);
                if(parser.isdataready()) {
                    xSemaphoreTake(xBin, 10);

```

```

        printf("Latitude: %f, Longitude: %f\r\n", parser.getLatitude(), parser.getLongitude());
        xSemaphoreGive(xBin);
        end = true;
    } else {
        xSemaphoreTake(xBin, 10);
        printf("NO SIGNAL\r\n");
        xSemaphoreGive(xBin);
        vTaskDelay(configTICK_RATE_HZ / 2);
    }
}
vTaskDelay(configTICK_RATE_HZ / 2);
}

/*
// GPRS/GSM 3G task (not in use)
static void vGSMTask(void *pvParameters) {
    GPRS gsm;
    char rx[64] = {0};
    char tx[25] = {0};
    int rc = 0;
    int tc = 0;
    while(1) {
        int t = Board_UARTGetChar();
        if (t != EOF) {
            xSemaphoreTake(xBin, 10);
            Board_UARTPutChar(t);
            xSemaphoreGive(xBin);
            tx[tc++] = toupper(t);
            if (t == '\r') {
                gsm.write(tx, tc);
                for(int i = 0; i < tc; i++) {
                    tx[i] = 0;
                }
                tc = 0;
                xSemaphoreTake(xBin, 10);
                printf("\r\n");
                xSemaphoreGive(xBin);
            }
        } else {
            vTaskDelay(1);
        }
        if(gsm.available()) {
            int r = gsm.read();
            if(r != -1) {
                if(rc < 63) {
                    rx[rc++] = r;
                    rx[rc] = 0;
                }
                if (r == '\n' || rc == 63) {
                    xSemaphoreTake(xBin, 10);
                    printf(rx);
                    xSemaphoreGive(xBin);
                    for(int i = 0; i < rc; i++) {
                        rx[i] = 0;
                    }
                    rc = 0;
                }
            }
        }
    }
}

```

```
// OBD task TODO: whole task
static void vOBDDTask(void *pvParameters) {
    while(1) {
        vTaskDelay(configTICK_RATE_HZ);
    }
}
*/

int main(void) {
    prvSetupHardware();
    xBin = xSemaphoreCreateBinary();

    xTaskCreate(vMPUTask, "vMPUTask",
                configMINIMAL_STACK_SIZE + 256, NULL, (tskIDLE_PRIORITY + 1UL),
                (TaskHandle_t *) NULL);

    xTaskCreate(vGPSTask, "vGPSTask",
                configMINIMAL_STACK_SIZE + 320, NULL, (tskIDLE_PRIORITY + 1UL),
                (TaskHandle_t *) NULL);
    /*
    xTaskCreate(vGSMTask, "vGSMTask",
                configMINIMAL_STACK_SIZE + 320, NULL, (tskIDLE_PRIORITY + 1UL),
                (TaskHandle_t *) NULL);

    xTaskCreate(vOBDDTask, "vOBDDTask",
                configMINIMAL_STACK_SIZE + 126, NULL, (tskIDLE_PRIORITY + 1UL),
                (TaskHandle_t *) NULL);
    */
    vTaskStartScheduler();

    return 1;
}
```