

Elcykelpool - Utvecklingsprojekt

Individuell rapport

av

Johannes Hellgren

Högskolan i Halmstad

Sammanfattning

Följande rapport beskriver utvecklingen av en elcykelpool. Systemet omfamnar en hemsida för uthyrning, elcyklar och ett garage för in-/utlämning av cykel. Med ett knapptryck så hissas cyklarna upp i garaget där de laddas automatiskt och förvaras säkert under skydd och låst dörr.

Innehållsförteckning

Sammanfattning	1
1. Inledning	3
1.1 Bakgrund	3
1.2 Syfte och mål	3
1.3 Avgränsningar	4
1.4 Problemställning	4
2. Teori	5
2.1 Utvecklingsmiljö	5
2.2 Datakommunikation	5
2.2.1 RS-485	5
2.2.2 Modbus RTU	6
2.2.3 UDP	8
3. Metod	9
3.1 Planering och förstudie	9
3.1.1 Utvärdering av tidigare arbete	10
3.1.2 Val av komponenter	10
3.2 Utveckling av delsystem	10
3.2.1 server, hemsida och databas	11
3.2.2 Gateway	12
3.2.3 Com-Module	13
4. Resultat	14
4.2 Hemsida	14
4.2 Server-Garage konversation	15
5. Diskussion	16
6. Referenser	17
7. Bilagor	19
7.1 Kravspecifikation	19
7.2 Projektplan	34
7.3 Tidsrapportering	46
7.4 Server : Databas	48

7.5 Server : UDP	50
7.5 Com-Module : Modbus	51

1. Inledning

Det här projektet går ut på att driftsätta en elcykelpool för studenterna på Högskolan i Halmstad.

1.1 Bakgrund

Projektet är baserat på det tidigare samarbetsprojektet Elmob mellan företagen Off Course och Move About som initierades av Göteborgs Stad. Elmob gick ut på att underlätta hållbart resande inom Göteborg med hjälp av elektriska transportlösningar. I och med att Off Course gick i konkurs så lades projektet Elmob ned och Move About fortsatte med sin elbilspool. Högskolan i Halmstad fick överta det färdiga materialet från Elmob.

Garaget är en modulkonstruktion där flera moduler kan anslutas till beroende hur stort garage man behöver, se figur 4 för systemöversikt. Varje modul har tre stycken fack. Varje fack har plats för en cykel, samt innehåller en dörr med lås, en hiss med knappsats, IR-sändtagare, olika sensorer, cykelladdare och ett "com-module" kretskort. Detta kretskort styrs alla delar i facket, sådant som lås, laddning och hiss. Kretskortet kommunicerar med BeagleBone-datorn via gränssnittet RS485. Den första modulen i garaget behöver vara en "huvudmodul" som utöver facken även innehåller ett elskåp som levererar ström till alla anslutna fack. Huvudmodulen innehåller också en BeagleBone Black, vilket är en enkortsdator som agerar som en "gateway" och sköter kommunikationen mellan varje com-module i de olika facken och servern.

Cyklarna är färdiga elcyklar från Off Course. När elcykeln befinner sig i garaget så kommunicerar den med com-module kretskortet via en IR-sändtagare. IR-sändtagaren kommer då att skicka information från de olika sensorer som sitter på cykel, samt berätta vilken cykel som är inlämnad. Cyklarna är färdiga system som vi kommer att använda men ej kommer att ändra på i vårt projekt.

1.2 Syfte och mål

Uppgiften med detta projekt är att driftsätta en tjänst för cykeluthyrning genom att använda oss av det tidigare materialet från Elmob. Detta innebär att förstå sig på och identifiera de delar av Elmob som vi vill ha kvar, ändra på eller kassera. Projekt är uppdelat i tre stycken delsystem, server, garage och cykel. Målet är att som student kunna logga in och hyra en elcykel via en hemsida på servern. Om det finns en ledig cykel så skall servern ge en signal till garaget att göra det möjligt för studenten att hämta ut en cykel. Det färdiga materialet från Elmob som detta arbete är baserat på omfamnar garage och cykel.

1.3 Avgränsningar

Servern har tidigare hanterats av Move About. Vi har inget tidigare arbete att fortsätta på, utan vi kommer att skapa en egen server. Servern kommer att bestå av en hemsida, samt olika databaser och PHP-skript[7].

Denna rapport kommer att fokusera på utveckling av gateway, server-kommunikation mot gateway, com-module kommunikation mot gateway samt com-module styrning av garage, se figur 4.

Se projektrapport[21] av Göran Trivic för djupare redovisning av utveckling av server med hemsida och databaser. Samt projektrapport[23] av André Kopfinger och projektrapport[22] av Niklas Eriksson för utveckling av garage och com-module.

1.4 Problemställning

- Server
 - Vad finns det för metoder för att skapa en server?
 - Vad finns det för metoder för att skapa en hemsida?
 - Vad finns det för metoder för att lagra data från hemsidan och garaget i databaser på servern?
- Kommunikation mellan server och garage
 - Hur kan servern hämta data och skicka styrsignaler till de olika facken i garaget?

2. Teori

2.1 Utvecklingsmiljö

LAMP[3] är ett mjukvarupaket för att utveckla och administrera servrar. Bland annat så innehåller LAMP programmet Apache HTTP Server för att skapa webbservrar, och phpMyAdmin[4] för att skapa och administrera MySQL[5] databaser.

Wordpress[9] är ett innehållshanteringssystem (CMS), och är ett av de populäraste publiceringssystemen för webbplatser. Från att Wordpress tidigare främst fokuserade på bloggsystem så har det utvecklats under åren och idag är en mycket kraftig tjänst för att redigera hemsidor med många olika syften. Det som gör tjänsten så kraftfull är dess tusentals olika tillägg (plugins), utvecklad av dess användare. Detta gör det möjligt att snabbt och enkelt få eftertraktade utseenden och funktioner på hemsidan med hjälp av dessa färdigutvecklade tillägg.

Raspberry Pi[2] och Arduino[10] är mikrokontrollerkort lämpade för arbeten där stora mängder processeringskraft ej är nödvändigt, och används bland annat inom Hemautomation, industriell automation, kommersiella produkter och många fler projekt där en liten dator behövs.

Raspberry Pi använder ett externt SD-kort som minne, där det Debian-baserade operativsystemet Raspbian[15] lagras. Den rekommenderade mjukvaran hämtas från raspberrys mjukvarupaket NOOBS[16]. Eftersom Raspberry Pi använder ett ”vanligt” operativsystem, kan valfritt programmeringsspråk och IDE med stöd för Debian användas.

Arduinos webbutvecklingsverktyg för Arduino Create[11] rekommenderas för programmering av Arduinos. Programmeringsspråket för webbutvecklingsverktyget är Arduinos egna programmeringsspråk, detta språk kallar olika C/C++ funktioner. Skripten kompileras sedan av en C/C++ kompilator (avr-g++).

2.2 Datakommunikation

2.2.1 RS-485

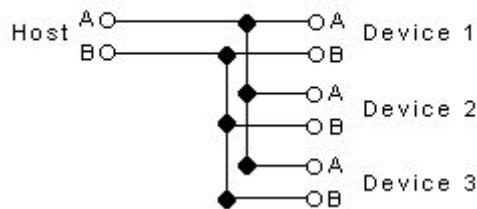
RS-485[8] är en elektrisk standard en för den fysiska kopplingen mellan master och slaves, ofta använd inom industriella kontrollsysteem såsom SCADA, DCS och PLCs.

RS485 är mycket vanligt och kan stödja upp till 32 apparater på ett avstånd upp till 1200 meter utan relästation.

Vid ihopkoppling så krävs det att en av datorerna är *Master* (max 1), och de resterande är *Slaves* (max 247) med unika slav-ID från 1 till 247. En slave kommer inte att skicka någon data förrän den får ett kommando från master att göra det. Master kan både skicka och läsa data från slave.

Vid datakommunikation mellan två mikrokontrollerkort via RS485 konverteras UART TTL seriell kommunikation till RS485 seriell kommunikation. När RS-485 *Two Wire* används så skickas den "vanliga" datan genom A(+), samtidigt som att samma data men inverterad skickas genom B(-), se figur 1. När meddelandet har anlånt så jämförs A och B för att kontrollera att inga störningar har påverkat meddelandet.

Typical 2-Wire RS-485 Wiring



Figur 1. RS485 koppling mellan master (host) och slaves (device 1,2,3). [18]

Datorerna som kommunicerar med varandra kräver att de har samma *Port settings* för att kommunikationen skall fungera. Inställningarna som behövs är *Baud rate* (96000 - 19200 är vanligast), *number of data bits* (7 eller 8), *number of stop bits* (1 eller 2) och *parity* (none, odd, even).

2.2.2 Modbus RTU

Modbus RTU är ett seriellt protokoll som följer master-slave strukturen och skickas vanligast över RS-485. Protokollet är enkelt att använda, mycket pålitligt och används ofta inom Industriella automationssystem (IAS).

Modbus RTU meddelanden använder en enkel 16 bitars CRC (Cyclic-Redundant Checksum) för att garantera pålitlighet. Se Modbus Protocol Reference[13] för beräkning av CRC.

Meddelandets struktur (se tabell 1) är baserat på kommunikation med PLCs, dessa har två outputs (en diskret och en analog) samt två inputs (en diskret och en analog), se tabell 2. ID:t avgör vilken slave meddelandet ska skickas till, funktionskoden (se tabell 3) berättar vad som skall göras med datan.

Adress (ID)	Funktion	Data	CRC
8 bitar	8 bitar	N * 8 bitar	16 bitar

Tabell 1. Modbus meddelande struktur

Minnesadress	Namn	Tillgång	Storlek

00001 - 09999	Coils (Discrete Outputs)	Read/Write	1 Bit
10001 - 19999	Inputs (Discrete Inputs)	Read Only	1 Bit
30001 - 39999	Input Registers (Analog Inputs)	Read Only	16 Bit
40001 - 49999	Holding Registers (Analog Outputs)	Read/Write	16 Bit

Tabell 2. Modbus minnesområde

Kod	Beskrivning	Funktion
01	Read coil status	Read
02	Read input status	Read
03	Read holding registers	Read
04	Read input registers	Read
05	Force single coil	Write
06	Preset single register	Write

Tabell 3. Modbus funktionskoder

Nedan (se figur 2) visas konversation mellan en master och slave. Master skickar en förfrågan (query) till slave 11, där master vill ha status på 25 st coils, börjat på adress 0x13.

QUERY	
Field Name	Example (Hex)
Slave Address	11
Function	01
Starting Address Hi	00
Starting Address Lo	13
No. of Points Hi	00
No. of Points Lo	25
Error Check (LRC or CRC)	—

Figur 2. Read Coil Status Query exempel. [13]

Nedan (se figur 3) visas svaret, där ID och funktion upprepas, men datan ersätts med dess värden på efterfrågade coils.

RESPONSE	
Field Name	Example (Hex)
Slave Address	11
Function	01
Byte Count	05
Data (Coils 27–20)	CD
Data (Coils 35–28)	6B
Data (Coils 43–36)	B2
Data (Coils 51–44)	0E
Data (Coils 56–52)	1B
Error Check (LRC or CRC)	—

Figur 3. Read Coil Status Response exempel. [13]

2.2.3 UDP

User Datagram Protocol(UDP)[17] är ett förbindelselöst protokoll för att skicka datagram över ett IP-nätverk. Eftersom protokollet är förbindelselöst så upprättar protokollet ej någon session mellan avsändaren och mottagaren, vilket innebär att sändaren inte kan garantera att en mottagare tar emot paketet som skickas. Mottagaren får heller inte veta att den fått rätt antal paket eller i rätt ordning. På grund av denna felmarginal på om paketen kommer fram, och i vilken ordning så är det viktigt att protokollet tillämpas på den områden där det är acceptabelt. UDP är dock en snabb form av kommunikation med betydligt mindre paketstorlek än andra ”pålitligare” protokoll, exempelvis TCP[17].

3. Metod

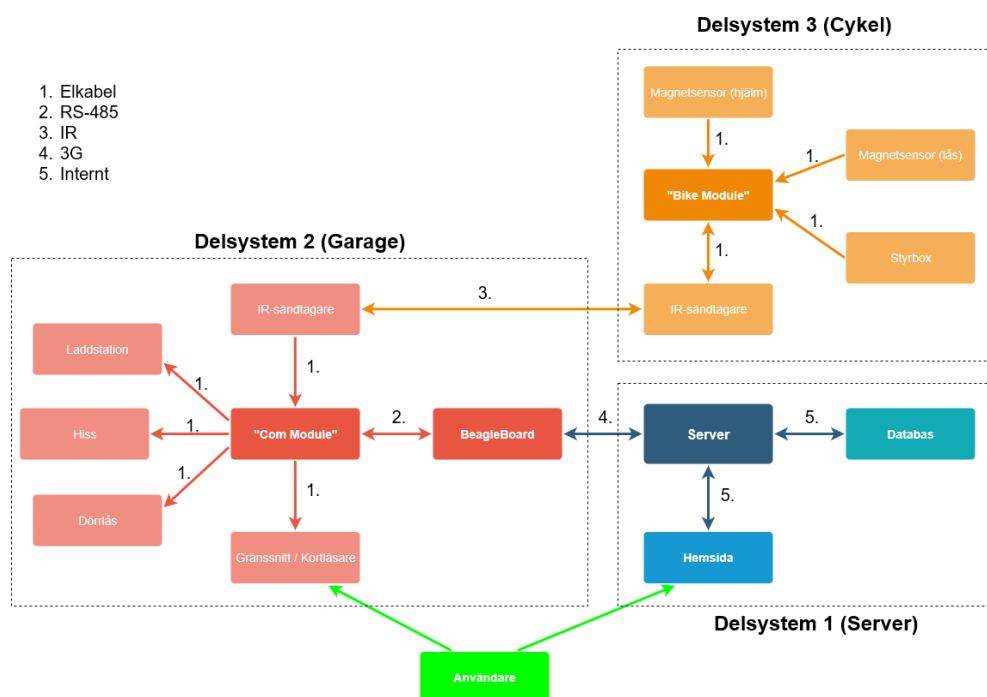
3.1 Planering och förstudie

Detta projekt har använt projektmodellen LIPS[25] som struktur för arbetets gång. Arbetet delades upp i tre faser: före, under och efter. Under hela arbetets gång har projektgruppen haft möte med handledare en gång i veckan för att gå igenom vad som gjorts och vad som skall göras.

Under den första fasen (före) togs en kravspecifikation [7.1](#) där vi delade upp projektet i tre delsystem och tilldelade arbetsområden för de olika gruppmedlemmar. Projektplan [7.2](#) och Tidplan med tidsrapportering [7.3](#) skapades också. En systemöversikt (se Figur 4) skapades för att få en överskådlig bild av systemet och hur de olika delsystemen interagerar med varandra. Efter detta så påbörjades förstudien, där gruppmedlemmarna läste på och tog fram nödvändig information för att utföra det tilldelade arbetet.

Under nästa fas (under) så påbörjades arbete på systemet. Både mjukvara och hårdvara utvecklas för att möta kraven i kravspecifikationen. Efter utvecklingen av systemet så utförs testerna i testspecifikationen. De enskilda systemen utvecklas och testas var för sig innan de kopplas ihop tillsammans för att få det färdiga systemet.

I den sista fasen (efter) så sker en demonstration av systemet samt en muntlig redovisning. En individuell rapport skrivas av alla gruppmedlemmar och dokumentationen lämnas in till handledare för bedömning och examination.



Figur 4. Systemöversikt

3.1.1 Utvärdering av tidigare arbete

Det tidigare arbetet utvecklas av AES gjordes väldigt hastigt utan någon dokumentation. Många delar saknades för att starta upp systemet, och de delar vi hade var skadade eller så krävde de utomstående system som vi ej hade tillgång till. Därför togs beslutet att utveckla en ny gateway och "com-module" från grunden, både mjukvara och hårdvara.

3.1.2 Val av komponenter

Eftersom det tidigare arbetet använde server skapad av MoveAbout så hade vi inte tillgång till det materialet. En Raspberry Pi 3 Model 3+ valdes att utveckla servern på.

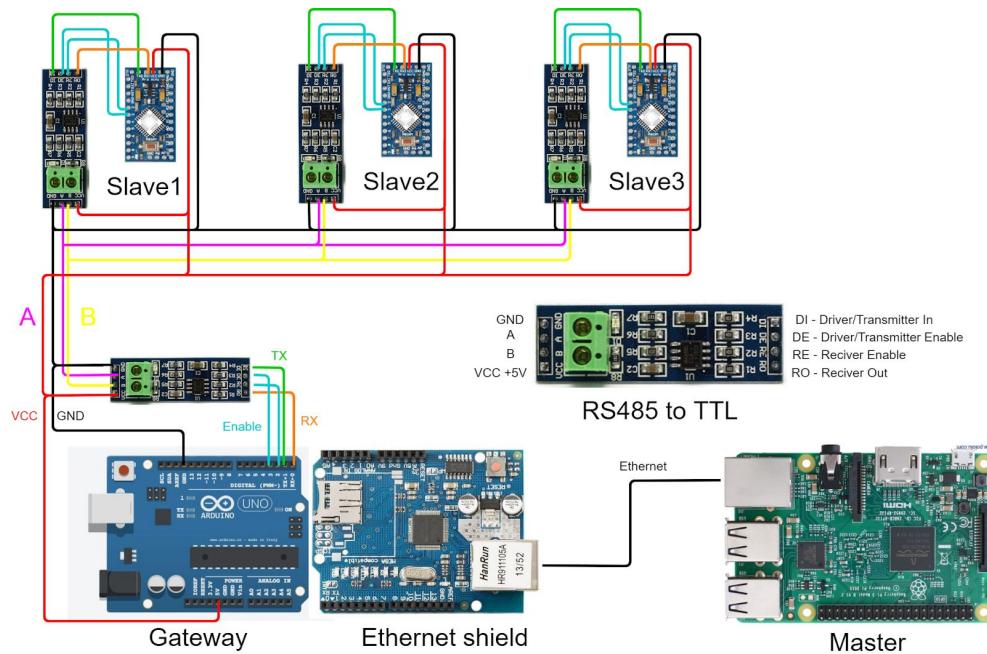
I form av en gateway så ersattes BeagleBone Black med en Arduino Uno. Till denna användes en ethernet-shield för att kunna skicka och ta emot UDP-paket till och från servern. Detta valdes för att göra det smidigt att arbeta med under utvecklingen av systemet. I praktiken är det tänkt att ett GSM nät skall användas, vilket inte påverkar kommunikationen då UDP-paketet kan skickas över både gränssnitten och övergången från ethernet till GSM eller Wi-Fi skall vara simpel.

Som mikrokontroller för com-module valdes Arduino Pro Mini. Detta gjorde att vi kunde programmera och testa kommunikationen för systemet innan kretskortet var tillverkat. Till skillnad från det gamla systemet som använder en fast mikroprocessor på kretskortet.

En TTL till RS-485 konverterar användes för att få hårdvaran som krävs för att skicka meddelanden över detta gränssnitt, både Arduino Uno och Arduino Pro Mini saknar denna kontakt.

3.2 Utveckling av delsystem

Efter att komponenterna hade valts så skapades ett kopplingsschema (se figur 5) för att få en överblick över hur de olika delarna kommunicerar med varandra, för att lära sig vilka kablar som behövs dras mellan stationerna, samt som en referens för ihopkoppling.



Figur 5. Koppling av master, gateway och slaves

Komponenterna kopplas först ihop på en breadboard för att få en "prototyp" av systemet enligt figur 5. Detta gjordes för att få ett lätt hanterbart system att arbeta med, samt för att säkerställa att rätt val av komponenter gjordes innan komponenterna monteras på den färdiga produkten.

3.2.1 server, hemsida och databas

LAMP användes på en Raspberry Pi 3 model B+ för att skapa servern. Databaserna hanteras genom phpMyAdmin. När hemsidan skapades med Wordpress så skapas mySQL databaser automatiskt av Wordpress för att hantera hemsidan, bland annat så sparas informationen om registrerade användare i dessa databaser eftersom ett Wordpress plugin användes för registrering. Informationen kan sedan hämtas med ett PHP skript, se [7.4 Server : Databas](#) för exempel.

Eftersom PHP används för att hämta och lägga in data i databaserna så skrevs också programmet för kommunikationen med garaget i PHP.

Modbus meddelanden skickas till garaget i form av UDP-paket över ethernet. Servern agerar som master. Kommunikationen med facken i garaget börjar med slave-ID 1 och fortsätter uppåt till antal slutna slaves, vilket i detta fallet går upp till slave-ID 3. Om en slave inte ger tillbaka något svar så återupprepas förfrågan upp till 10 gånger, innan programmet går vidare, se [7.5 Server : UDP](#).

Med jämna mellanrum så begär servern status på om cykeln laddas eller inte.

Om en förfrågan på att hämta ut en cykel uppstår i databasen så skickas en förfrågan till respektive com-module för att ”starta” facket.

3.2.2 Gateway

Kommunikationen mellan gateway och com-modules sker över *RS-485 Two Wire Half Duplex*.

Gateway har som uppgift att konvertera meddelanden från ett format till ett annat. I detta fall tar gateway emot UDP[17] meddelande över ethernet från servern och konverterar om det till ett seriellt meddelande över RS-485. Vice versa sker när det kommer ett meddelande från RS-485 länken som sedan skickas till servern som ett UDP-paket. Gateway läser därför inte av och behandlar någon information i meddelandet. Utan skickar endast vidare informationen.

För att ta emot UDP-paket över ethernet används en ethernet-shield för att få den fysiska kontakten till Arduino Uno, och Arduinos bibliotek `<Ethernet.h>` och `<EthernetUdp.h>` för att ta emot och skicka paketen. På liknande sätt används en TTL till RS-485 konverterare för att få den fysiska kontakten för RS-485 till Arduino Uno, och Arduinos bibliotek `<ArduinoRS485.h>` för att ta emot och skicka meddelanden. Detta bibliotek tar ej hänsyn till den fysiska konverteraren. Därför krävs det att konverteraren konfigureras beroende på om den skall skicka eller ta emot meddelanden.

Meddelandet som går mellan server och de olika com-module korten är ett modbus meddelande. Meddelandet är skapat efter *Modicon Modbus Protocol Reference Guide*[13].

Modbus protokollet som utvecklades använder överföringskontrolltecken[12] SOH (Start of Heading, 0x01) och EOT (End of Transmission, 0x04) istället för tomgångsperioder för att avgöra början och slut på meddelanden över RS-485. Dessa tecken är samma tecken som används av UDP meddelanden mellan server och gateway. Detta gör dem lämpliga att också använda mellan gateway och com-module slaves, eftersom meddelandet inte behöver ändra utseende genom passage av gateway.

Gateway använder ej DHCP, utan konfigureras med den statiska IP-adressen 192.168.1.177, MAC-adressen 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED, och använder port nr. 8888 för UDP kommunikation.

3.2.3 Com-Module

Som del av ”com-module” används Arduino Pro Mini för att ta emot och skicka Modbus RS-485 meddelanden till servern, och för att styra och läsa av kretskortet som mikrokontrollern är kopplat till. Se [7.5 Com-Module : Modbus](#) för exempel på mottagning och svar på modbus meddelande.

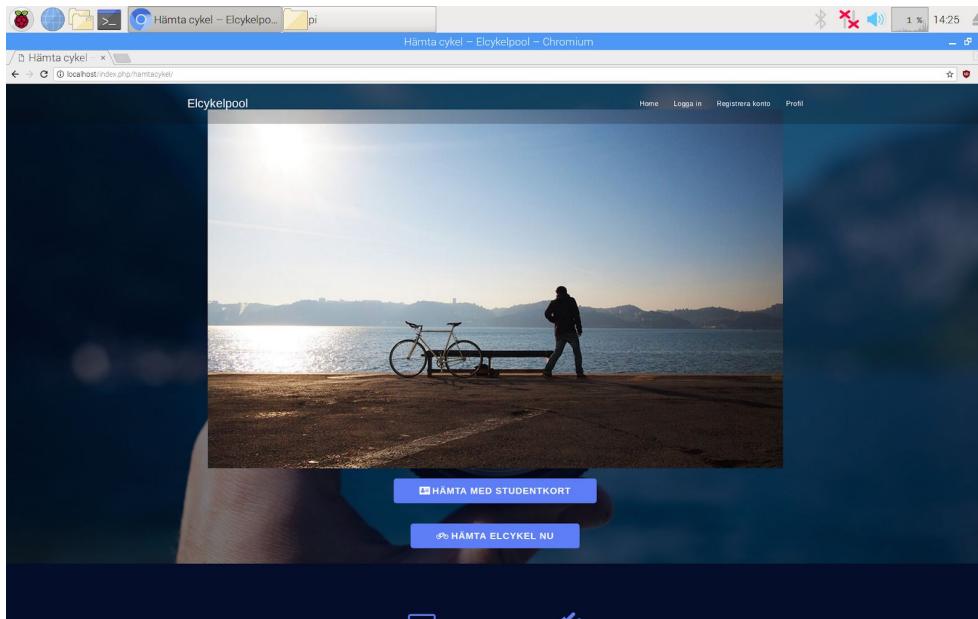
Huvuduppgifterna för mikrokontrollern är att ”starta” facket, samt att skicka status på facket tillbaka till servern. När facket har ”startat” så skall bland annat lampor tändas, dörren ska låsas upp och cykeln ska kunna hissas upp och ner genom att trycka på knappsatsen, med andra ord skall en kund kunna hämta ut en cykel samt lämna in en cykel.

8 st coils skapades med startaddress 0x3E, eftersom denna del av minnet inte används av processorn och ej riskerar att överskridas. Register GPIOR0 (0x3E), GPIO1 (0x4A), GPIO2 (0x4B) [26] användes under utveckling och tester för samma ändamål.

4. Resultat

4.2 Hemsida

Hemsida vid bokning av cykel.



Figur 6. Hemsida

Databaser administration från phpMyADmin. Olika tabeller visas till vänster. Innehåll av ”Cykel_Fack” tabell med innehållet fack-ID, cykel-ID och tillgänglighet för uthyrning av cykel.

 A screenshot of the phpMyAdmin interface. On the left, a sidebar lists various databases and tables, including 'Cykel_Fack'. The main area shows the contents of the 'Cykel_Fack' table with three rows of data:

	fackID	cykelID	available	lastupdated
1	1	1	1	2018-12-21 14:21:18
2	2	2	0	2018-12-22 16:30:48
3	3	3	0	2018-12-22 16:28:39

Figur 7. Databaser

4.2 Server-Garage konversation

Tre stycken förfrågan från server om att först läsa status på 8 st coils, börjat från address 0x3E, ändra första värdet på coil vid address 0x3E, upprepad första förfrågan för att se förändringen.

```
Slave 1 Startup

Print buffer:
1 1 1 0 3E 0 8 4
Sending response
11180000000004

Print buffer:
1 1 5 0 3E 0 1 4
Set to HIGH
Sending response
11503E014

Print buffer:
1 1 1 0 3E 0 8 4
Sending response
11181000000004
```

Figur 8. Server-Garage konversation från com-module

Samma konversation som ovan från serverns synpunkt. Styrning av garage samt avläsning av status.

```
pi@raspberrypi:~ $ php /home/pi/PhpModbus/phpmodbus-master/UDPClient.php
UDP Client Startup
Socket created
Socket bind OK
Sending readCoils
Time: %Sat, 29 Dec 2018 20:55:18 +0100
Received response from:
192.168.1.177 : 8888 -- 0105003e0001

pi@raspberrypi:~ $ php /home/pi/PhpModbus/phpmodbus-master/UDPClient.php
UDP Client Startup
Socket created
Socket bind OK
Sending readCoils
Time: %Sat, 29 Dec 2018 20:50:39 +0100
Received response from:
192.168.1.177 : 8888 -- 010108000000000000000000

pi@raspberrypi:~ $ php /home/pi/PhpModbus/phpmodbus-master/UDPClient.php
UDP Client Startup
Socket created
Socket bind OK
Sending readCoils
Time: %Sat, 29 Dec 2018 20:56:55 +0100
Received response from:
192.168.1.177 : 8888 -- 010108010000000000000000
```

Figur 9. Server-Garage konversation från server

5. Diskussion

Eftersom beslutet gjordes att kasta det gamla arbetet som gjordes på gateway och com-module så fanns risken att inte hinna få ett fungerande system på grund av tidsbrist. Dock så blev slutresultatet bättre än förväntat. Både mjukvara och hårdvara blev färdigställd och grundkraven från kravspecifikationen [7.1 Kravspecifikation](#) uppfylldes.

Nu i efterhand känns det bra att vi gjorde om delar av systemet från grunden. Detta gav oss djupare förståelse i hur kommunikationen mellan de olika delarna sker, och vi blev tvungna att lära oss och utveckla protokoll och turordningsregler för kommunikation.

Arbetet påbörjades betydligt senare än planerat. Detta var på grund av problem med förflyttningen av garaget. Förstudieperioden blev därmed väldigt lång, näst intill halva terminen, men eftersom vi ej hade tillgång till materialet eller kunnat titta på garaget så kunde endast en begränsad förstudie göras eftersom vi inte visste vad för komponenter som satt på garaget. Efter att garaget var flyttat och vi fick tillgång till materialet så kunde vi påbörja utvärdering av det tidigare arbetet och göra upp en plan om vad som behövs göras.

6. Referenser

- [1] "Modbus interface tutorial," [Online]. Available: <https://www.lammertbies.nl/comm/info/modbus.html> . [Använd 15 November 2018].
- [2] "Raspberry pi 3 model B," [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/> . [Använd 14 November 2018].
- [3] "LAMP," [Online]. Available: [https://en.wikipedia.org/wiki/LAMP_\(software_bundle\)](https://en.wikipedia.org/wiki/LAMP_(software_bundle)). [Använd 3 December 2018].
- [4] "phpMyAdmin," [Online]. Available: <https://en.wikipedia.org/wiki/PhpMyAdmin>. [Använd 2 Oktober 2018].
- [5] "MySQL," [Online]. Available: <https://en.wikipedia.org/wiki/MySQL>. [Använd 4 Oktober 2018].
- [6] "Visual Studio Code," [Online]. Available: https://en.wikipedia.org/wiki/Visual_Studio_Code. [Använd 17 Oktober 2018].
- [7] "PHP," [Online]. Available: <https://en.wikipedia.org/wiki/PHP>. [Använd 17 Oktober 2018].
- [8] "RS485," [Online]. Available: <https://en.wikipedia.org/wiki/RS-485>. [Använd 17 Oktober 2018].
- [9] "Wordpress," [Online]. Available: <https://en.wikipedia.org/wiki/WordPress>. [Använd 17 Oktober 2018].
- [10] "Arduino Uno," [Online]. Available: https://en.wikipedia.org/wiki/Arduino_Uino. [Använd 17 Oktober 2018].
- [11] "Arduino Create," [Online]. Available: <https://www.arduino.cc/en/Main/Create>. [Använd 17 Oktober 2018].
- [12] "Överföringskontrolltecken," [Online]. Available: https://en.wikipedia.org/wiki/Control_character. [Använd 29 December 2018].
- [13] "Modicon Modbus Protocol Reference Guide," [Online]. Available: http://modbus.org/docs/PI_MBUS_300.pdf. [Använd 17 Oktober 2018].

- [14] "Arduino Pro Mini," [Online]. Available:
https://wiki.eprolabs.com/index.php?title=Arduino_Pro_Mini. [Använt 03 Januari 2018].
- [15] "Raspian," [Online]. Available:
<https://www.raspberrypi.org/downloads/raspbian/>. [Använt 03 Januari 2018].
- [16] "NOOBS," [Online]. Available:
<https://www.raspberrypi.org/downloads/noobs/>. [Använt 17 Oktober 2018].
- [17] "TCP & UDP," [Online]. Available:
<https://www.iis.se/lar-dig-mer/guider/introduktion-till-ip-internet-protocol/tcp-och-udp-nivan/>. [Använt 17 Oktober 2018].
- [18] "RS-485 Wiring," [Online]. Available:
<https://www.omega.co.uk/techref/das/rs-232-422-485.html>. [Använt 17 Oktober 2018].
- [21] Göran Trivic, "Projektrapport," [Online]. Available:
<https://drive.google.com/drive/folders/1-Taw-cmUlf9lV35n3Ayy5mJ2CngsCyEg>. [Använt 07 Januari 2018].
- [22] Niklas Eriksson, "Projektrapport," [Online]. Available:
<https://drive.google.com/drive/folders/1-Taw-cmUlf9lV35n3Ayy5mJ2CngsCyEg>. [Använt 07 Januari 2018].
- [23] André Kopfinger, "Projektrapport," [Online]. Available:
<https://drive.google.com/drive/folders/1-Taw-cmUlf9lV35n3Ayy5mJ2CngsCyEg>. [Använt 07 Januari 2018].
- [24] "Modbus meddelande exempel," [Online]. Available:
<https://www.dghcorp.com/modbus/modbusrtu05.asp>. [Använt 09 Januari 2018].
- [25] Tomas Svensson Christian Krysander. Projektmodellen LIPS. 2011.
- [26] "ATmega328P Data Sheet," [Online]. Available:
<http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf>. [Använt 09 Januari 2018].

7. Bilagor

7.1 Kravspecifikation

KRAVSPECIFIKATION

ELCYKELPOOL

Johannes Hellgren, Göran Trivic, André Kopfinger, Niklas Eriksson

Status

Granskad	Hans-Erik Eldemark	2018-10-16
Godkänd	Hans-Erik Eldemark	2011-10-xx



Elcykelpool

HT18, Elcykelpool
Högskolan i Halmstad, institution

Namn	Ansvar	Telefon	E-post
Johannes Hellgren	projektledare (mek)	072-302 53 54	johhel16@student.hh.se
Göran Trivic	server/databas (dat)	070-750 30 91	Gortri16@student.hh.se
Niklas Eriksson	garage & cykel (elt)	072-547 47 53	Nikeri16@student.hh.se
André Kopfinger	garage & cykel (elt)	076-631 61 91	Andkop16@student.hh.se

E-postlista för hela gruppen: johhel16@student.hh.se

Kursansvarig: Nicolina Måansson, E304, 035-16 7487, nicolina.mansson@hh.se

Handledare: Hans-Erik Eldemark, 035-16 7203., 070 844 28 38., Hans-Erik.Eldemark@hh.se

**Innehåll**

0 Dokumenthistorik	5
1 INLEDNING	6
1.1 Parter	6
1.2 Syfte och Mål	6
1.3 Användning	6
1.4 Bakgrundsinformation	7
1.5 Definitioner	7
2 ÖVERSIKT AV SYSTEMET	8
2.1 Grov beskrivning av produkten	8
2.2 Produktkomponenter	8
2.3 Beroenden till andra system	8
2.4 Ingående delsystem	9
2.5 Avgränsningar	9
2.6 Designfilosofi	9
2.7 Generella krav på hela systemet	9
3 DELSYSTEM 1: SERVER	9
3.1 Inledande beskrivning av delsystem 1	9
3.2 Gränssnitt	10
3.3 Designkrav	10
3.4 Funktionella krav för delsystem 1	10
4 DELSYSTEM 2: GARAGE	11
4.1 Inledande beskrivning av delsystem 2	11
4.2 Gränssnitt	11
4.3 Designkrav	12
4.4 Funktionella krav för delsystem 2	12
5 DELSYSTEM 3: CYKEL	12
5.1 Inledande beskrivning av delsystem 3	13
5.2 Designkrav	13
5.3 Funktionella krav för delsystem 3	13
6 PRESTANDAKRAV	14
7 KRAV PÅ VIDAREUTVECKLING	14
8 TILLFÖRLITLIGHET	14
9 EKONOMI	14
10 KRAV PÅ SÄKERHET	14



11 LEVERANSKRAV OCH DELLEVERANSER	14
12 DOKUMENTATION	14
13 KVALITETSKRAV	15
14 UNDERHÅLLBARHET	15
15 REFERENSER	15

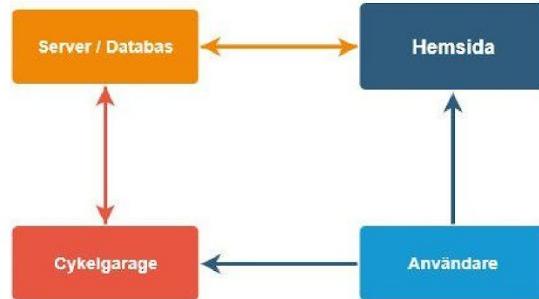
**0 Dokumenthistorik**

Version	Datum	Utförda förändringar	Utförda av	Granskad
0.01	2018-09-20	Första versionen	ckr	
0.2	2018-10-12	Första utkastet	ckr	
0.3	2018-10-17	Andra utkastet	ckr	



1 INLEDNING

Eleykel-uthyrningssystem, där studenter kan via en hemsida hyra en elcykel som sedan hämtas ut och lämnas tillbaka i ett cykelgarage. Se figur 1.



Figur 1. Systemet i dess omgivning.

I detta dokument finns alla krav listade genom en tabellrad likt den som ses nedan. Kravnummer löper genom hela dokumentet. Kolumn 2 visar om kravet är ett original krav eller om kravet har förändrats. Om det har förändrats så visas en referens till beslutet. I kolumn 3 finns beskrivningen av kravet. I den sista kolumnen visas prioriteten för kravet. Prioritet 1 är minimikravet. Prioritet 2 är tillägg och optimeringar för att förbättra och modernisera systemet. Prioritet 3 är framtidsvisionen.

Krav nr. x	Förändring	Kravtext för krav nr X	prioritet
------------	------------	------------------------	-----------

1.1 Parter

Producenterna i det här projektet är följande: Johannes Hellgren, Göran Trivic, André Kopfinger, Niklas Eriksson. Vi har valt att specificera studenterna på Högskolan i Halmstad som användare.

1.2 Syfte och Mål

Syftet med projektet är att bidra till en bättre miljö, samt göra en tjänst för campus genom att underlätta för dess studenter att ta sig till olika platser.

Målet med projektet är att få ett fungerande system som är anpassat för Halmstads studenter.

1.3 Användning

Grundtanke är att projektet ska kunna användas av studenter för att miljövänligt och smidigt kunna transportera sig till och från högskolan.



1.4 Bakgrundsinformation

Projektet kallades Elmob från början och utvecklades tillsammans med olika företag där MoveAbout och Off Course var de största faktorerna för utvecklingen av elcykelpoolen. Under processen av utvecklingen gick Off Course i konkurs och det slutade med att Halmstads Högskola fick ta över det som var kvar med projektet. I nuläget arbetar vi vidare med detta projekt med våra egna idéer.

1.5 Definitioner

Radio-frequency identification (RFID) är en teknik för att läsa information på avstånd från transpondrar och minnen, Dessa kallas för taggar.

Global positioning system (GPS) är ett system för satellitnavigering.

Kortvägigt infrarött (IR) ljus används för trådlös informationsöverföring.

Ett garage(modul) är en skyddad byggnad med 3 fack, 3 dörrar, ett huvudelskåp, samt ett gränssnitt för uthämtning av elcykel. Garaget innehåller också en Arduino

En Arduino & BeagleBone är en enkortsdator.

Ett fack i garaget är en hållplats för en cykel. Facket innehåller en hiss, IR sändtagare, en laddstation och en Com-Module.

”Com-Module” är kretskortet och datorn som styr alla komponenter i facken, samt skickar och tar emot information mellan sig själv och vår gateway.

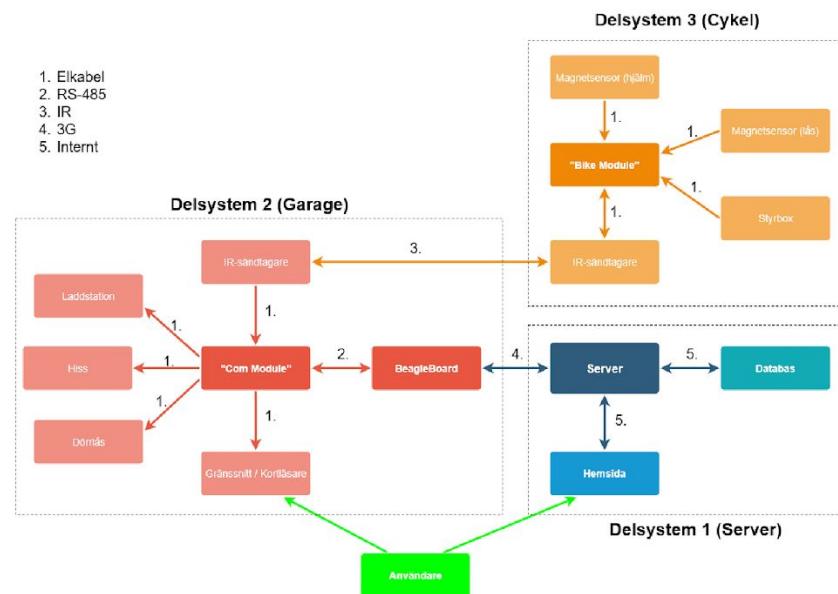
Gateway är en Arduino som tar emot och skickar vidare information mellan server och olika fack.

”Flaggade” cyklar är cyklar som anses vara ur funktion och som ej går att boka.



2 ÖVERSIKT AV SYSTEMET

Block-bild över hur de olika komponenterna samt delsystemen kommunicerar med varandra.
Delsystem 1: Server (blått), delsystem 2: Garage (rött), delsystem 3: cykel (gult). Se figur 2.



Figur 2. Denna bild visar en översikt av systemet.

2.1 Grov beskrivning av produkten

Användaren hyr en cykel via hemsidan och sedan hämtar och lämnar cykeln i garaget. Servern hanterar användarens beställning och debitering. Servern hämtar information om användaren, garaget och cyklarna som lagras i en databas. BeagleBoard-datorn styr utlämning och laddning av cyklarna. Cyklarna skickar sin position till servern via GPS, samt att de kommunicerar med garaget via IR länk när cykeln är tillbakalämmad.



2.2 Produktkomponenter

Garage med tillhörande komponenter, elcyklar, server med databas och hemsida, individuella rapporter samt övrig dokumentation.

2.3 Beroenden till andra system

2.4 Ingående delsystem

Delsystem 1 (Server) innehåller en hemsida, databas och intelligensen för uthyrning systemet. Delsystem 2 (Garage) innehåller en knappsats, dörr, hiss, laddstationer, huvudelskåp och en Arduino-enkortsdator. Delsystem 3 (Cykel) innehåller en IR länk.

2.5 Avgränsningar

Vi kommer endast att fokusera på driftsättning av en modul, samt uthyrningssystem för cyklar hos en modul. I verkligheten när alla moduler ska vara igång, ska varje fack prata med sitt kretskort som sedan pratar med en raspberry pi, arduino som vi kallar huvud arduino funkar som en gateway och sköter kommunikationen mellan fack och raspberry pi, t.ex. ifall de blir något systemfel i fack 4 skickas detta till raspberry pi att det är fel i just fack 4.

Vi kommer ta bort allt vi stöter på medans vi jobbar med projektet som vi känner är onödigt för att nå våra krav.

2.6 Designfilosofi

Vi utgår från ett befintligt garage och elcykel.

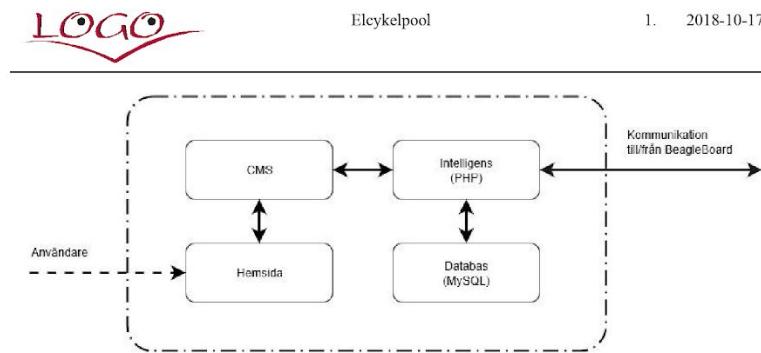
2.7 Generella krav på hela systemet

Nedan listas det kravet som är generellt för hela systemet.

Krav nr. 1	Original	Systemet innehåller ett cykelgarage för uthämtning & inlämning av elcykel.	Prioritet 1
Krav nr. 2	Original	Användare skall kunna använda systemet för att hyra en elcykel via en hemsida.	Prioritet 1

3 DELSYSTEM 1: SERVER

Hemsida som hanterar kund-registrering/login, uthyrning av cykel med debitering. Databas som lagrar kundinformation, cykel-id, batteritillstånd, cykelposition.



Figur 3. Bild av delsystem 1.

3.1 Inledande beskrivning av delsystem 1

Krav nr 1	Original	Systemet skall innehålla en hemsida för uthyrning av cykel.	Prioritet 1
Krav nr 2	Original	Systemet skall innehålla PHP skript som är intelligensen bakom uthyrningen.	Prioritet 1
Krav nr 3	Original	Systemet skall innehålla en databas för lagring av kunduppgifter, samt information om garaget och cyklarn.	Prioritet 1
Krav nr 4	Original	Användaren ska kunna registrera sig och logga in på hemsidan.	Prioritet 1

Krav nr 5	Original	Batterinivån och de cyklar som är bokbara skall framgå på hemsidan.	Prioritet 2
Krav nr 6	Original	Användaren skall vid registrering behöva skriva in X betalningsmetod för sin profil för att kunna hyra en cykel.	Prioritet 2
Krav nr 7	Original	Hemsidan ska visa en karta över Halmstad, där positionen för garaget visas samt antalet cyklar i garage.	Prioritet 3

Krav nr 8	Original	Användaren ska kunna hyra en cykel som inte är flaggad.	Prioritet 1
-----------	----------	---	-------------

3.2 Gränssnitt

Krav nr 9	Original	Servern skall kunna kommunicera med en Raspberry Pi via 3G.	Prioritet 2
-----------	----------	---	-------------

3.3 Designkrav



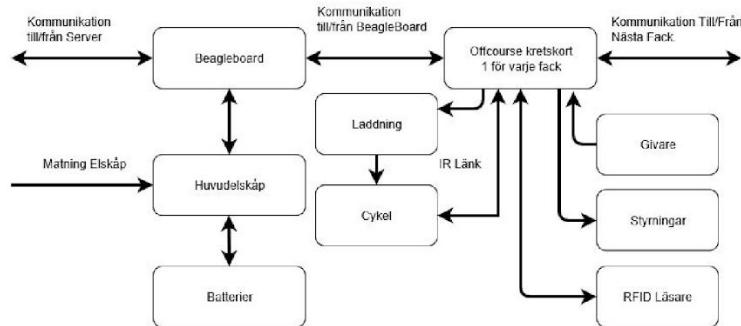
Krav nr 10	Original	Databas skall konstrueras med MySQL	Prioritet 1
Krav nr 11	Original	Hemsidan skall konstrueras med Wordpress	Prioritet 1

3.4 Funktionella krav för delsystem 1

Krav nr 12	Original	Cykeln skall endast vara bokbar om batterinivån överskrider X %	Prioritet 2
Krav nr 13	Original	Cyklar som anses vara ur funktion skall manuellt kunna flaggas och skall då inte kunna bli bokade.	Prioritet 3

Krav nr 14	Original	Hemsidan skall erbjuda en alternativ registrering och inloggning via Facebook och Google.	Prioritet 2
Krav nr 15	Original	Databasen ska innehålla användarens inloggningssuppgifter, student-id.	Prioritet 1
Krav nr 16	Original	Databasen skall innehålla hur många cyklar som finns tillgängliga.	Prioritet 2
Krav nr 17	Original	När servern får ett RFID-nummer från huvud arduinon så skall servern skicka tillbaka ifall numret finns i databasen eller inte.	Prioritet 2

4 DELSYSTEM 2: GARAGE



Figur 4. Översikt Garage

4.1 Inledande beskrivning av delsystem 2

Garaget består av en huvuddel som man sedan kan ansluta moduler till med fler fack.



Huvuddelen består av ett elskäp, en Raspberry pi 3 och 3 st fack där varje fack innehåller ett kretskort som styr funktionerna för varje fack. För varje fack finns det en hiss och en dörr, där alla fack ser exakt likadana ut.

Modulerna som man kan ansluta innehåller 3 st fack.

Krav nr 18	Original	Göra kopplingsschema för kretskortet	Prioritet 1
Krav nr 19	Original	Systemet skall ha en Gateway (Arduino Uno) som kommunicerar med servern via Ethernet.	Prioritet 1
Krav nr 20	Original	Gateway skall kommunicera med de olika com-modulerna (Arduino Pro Mini) i facken via RS485	Prioritet 1
<hr/>			
Krav nr 21	Original	Få igång styrning av lås i facken	Prioritet 1
<hr/>			
Krav nr 22	Original	Få igång styrning av cykelhiss i facken	Prioritet 1
Krav nr 23	Original	Få igång givare i facken.	Prioritet 1

4.2 Gränssnitt

Krav nr 24	Original	Gateway-datorn skall kommunicera med servern via 3G	Prioritet 2
Krav nr 25	Original	Gateway-datorn skall kommunicera med cyklarna via IR	Prioritet 3

4.3 Designkrav

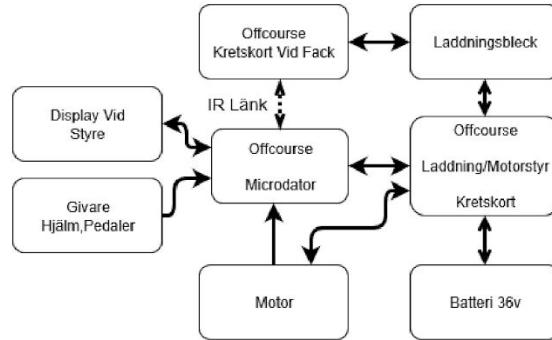
Krav nr 26	Original	Gateway-datorn skall programmeras med Arduinos Web Editor <i>Arduino Create</i> i språket C	Prioritet 2
------------	----------	--	-------------

4.4 Funktionella krav för delsystem 2

Krav nr 27	Original	Systemet ska kolla om cykeln laddas efter inlämning.	Prioritet 1
Krav nr 28	Original	Användaren skall kunna hissa upp och ner cykeln via knappslätsen.	Prioritet 1
Krav nr 29	Original	RFID läsare skall kunna läsa RFID tagg och skicka serienummer till Raspberry Pi -datorn	Prioritet 2
<hr/>			
Krav nr 30	Original	När användaren vill hämta ut en cykel via hemsidan så skall låset läsas upp.	Prioritet 1
Krav nr 31	Original	Systemet skall flagga cykeln om den inte får kontakt med laddbläckan.	Prioritet 2
Krav nr 32	Original	RFID tagg skall läsa samma nummer som det står på studentkorten.	Prioritet 2



5 DELSYSTEM 3: CYKEL



Figur 5. Översikt Cykel

5.1 Inledande beskrivning av delsystem 3

Cykelsystemet förutom själva cykeln, innehåller en microdator, ett laddning- och motorstyrkort, ett batteri ,en motor ,givare för hjälm och pedaler och en display vid cykelns styre.

Microdatorn som är från OffCourse är själva hjärnan på cykeln och den samlar information som nuvarande batteri, den totala sträckan som är körd, samt läser av de andra sensorerna på cykeln som magnet sensorn för hjälmen. Informationen skickas sedan till OffCourse kretskortet som sitter i facket via en IR länk som sitter på båda cykeln och facket. Facket stämmer sedan av så att allt är okej skulle det vara något fel som att hjälmen inte finns eller cykeln gick inte och ladda meddelas systemet att något är fel.

Offcourse laddning och motorstyrnings kretskort har hand om laddning respektive styrningen av motorn, och sitter på samma kretskort. Kortet har koll på laddningsnivån på batterierna och motorstyrningsdelen sköter drivningen av motorn.

Där finns givare som kollar om hjälmen är på plats och om pedalaerna används så att motorn kan driva.

Batteriet har en spänning på 36V.

Finns även en display vid styret som visar batterikapacitet och hastighet.

Krav nr 33	Original	Skall förbättra säkerheten på cykeln	Prioritet 2
Krav nr 34	Original	Kommunikation mellan cykel och garaget via IR skall fungera.	Prioritet 2
Krav nr 35	Original	Sätta fast kontakt-blocket på cykeln igen.	Prioritet 1



Krav nr 36	Original	Två av kontaktbleckena skall användas för att känna av om cykeln är upphissad eller inte.	Prioritet 1
------------	----------	---	-------------

5.2 Designkrav

Krav nr 37	Original	Gps på batteriet för ökad säkerhet	Prioritet 3
------------	----------	------------------------------------	-------------

5.3 Funktionella krav för delsystem 3

Krav nr 38	Original	Kommunikation mellan cykel batteri och server när cykeln används.	Prioritet 2
Krav nr 39	Original	RFID-tag i hjälm, preliminärt cykel.	Prioritet 2
Krav nr 40	Original	Batteriet ska automatiskt laddas efter inlämning.	Prioritet 1

6 PRESTANDAKRAV

Krav nr 41	Original	Cykeln ska kunna färdas 5 mil på en laddning.	Prioritet 2
------------	----------	---	-------------

7 KRAV PÅ VIDAREUTVECKLING

Krav nr 42	Original	Vid vidare utveckling krävs de att systemet ska kunna anpassas för olika elcyklar då de endast finns ett fåtal cyklar kvar som detta system kan hantera. Ifall fler garage byggs krävs det att systemet ska kunna kommunicera mellan de olika garagen för att observera så det alltid finns cyklar/utrymme.	Prioritet 3
------------	----------	--	-------------

8 TILLFÖRLITLIGHET

Krav nr 43	Original	Få igång reservbatterier.	Prioritet 2
------------	----------	---------------------------	-------------

9 EKONOMI



Krav nr 44	Original	Projektets arbetsstimmor ska ligga runt 1600 timmar där timmarna delas upp för varje medlem. Vilket leder till 400 timmar per medlem.	Prioritet 1
------------	----------	---	-------------

Krav nr 45	Original	Projektets budget är 3000kr.	Prioritet 1
------------	----------	------------------------------	-------------

10 KRAV PÅ SÄKERHET

Krav nr 46	Original	Prioritet 1
------------	----------	-------------

11 LEVERANSKRAV OCH DELLEVERANSER

Krav nr 47	Original	Prioritet 1
------------	----------	-------------

12 DOKUMENTATION

Dokument	Språk	Syfte	Målgrupp	Format/ media
Elschema	Svenska	Beskriver elsystemet	Tekniska ansvariga	Elektroniskt.

13 KVALITETSKRAV

Krav nr 48	Original	Prioritet 1
------------	----------	-------------

14 UNDERHÅLLBARHET

Krav nr 49	Original	Prioritet 1
------------	----------	-------------

15 REFERENSER

7.2 Projektplan

PROJEKTPLAN

ELCYKELPOOL

Johannes Hellgren, Göran Trivic, André Kopfinger, Niklas Eriksson



LOGO

2018-10-17

Status

Granskad	Hans-Erik Eldemark	2018-10-17
Godkänd	Hans-Erik Eldemark	2018-10-

2018-10-17

PROJEKTIDENTITET

Elcykelpool 2018/09-01
Halmstad Högskola

Namn	Ansvar	Telefon	E-post
Johannes Hellgren	projektledare (mek)	072-302 53 54	johhel16@student.hh.se
Göran Trivic	server(dat)	070-750 30 91	Gortri16@student.hh.se
Niklas Eriksson	garage & cykel (elt)	072-547 47 53	Nikeri16@student.hh.se
André Kopfinger	garage & cykel (elt)	076-631 61 91	Andkop16@student.hh.se

E-postlista för hela gruppen: johhel16@student.hh.se

Kursansvarig: Nicolina Måansson, E304, 035-167487, nicolina.mansson@hh.se
 Handledare: Hans-Erik Eldemark, 035-167203, 070-8442838, hans-erik.eldemark@hh.se



2018-10-17

Innehåll

Dokumenthistorik	6
BESTÄLLARE	6
ÖVERSIKTLIG BESKRIVNING AV PROJEKTET	6
Syfte och mål	6
Leveranser	6
Begränsningar	6
FASPLAN	6
Före projektstart	6
Under projektet	6
Efter projektet	7
ORGANISATIONSPLAN FÖR HELA PROJEKTET	7
Villkor för samarbetet inom projektgruppen	7
Definition av arbetsinnehåll och ansvar	7
DOKUMENTPLAN	8
UTBILDNINGSPLAN	8
Egen utbildning	8
RAPPORTERINGSPLAN	8
MÖTESPLAN	9
RESURSPLAN	9
Personer	9
Material	9
Lokaler	9
Ekonomi	9
MILSTOLPAR OCH BESLUTSPUNKTER	9
Milstolpar	9
Beslutspunkter	10
AKTIVITETER	10
TIDPLAN	10
FÖRÄNDRINGSPLAN	11
KVALITETSPLAN	11
Granskningar	11
Testplan	11
RISKANALYS	11

Utvecklingsprojekt
LIPS Projektplan

4

Elcykelpool
johhe16@student.hh.se



2018-10-17

PRIORITERINGAR

11

PROJEKTAVSLUT

12



2018-10-17

1 DOKUMENTHISTORIK

Version	Datum	Utförda förändringar	Utförda av	Granskad
1.0	2018-10-17	Första versionen	ckr	

2 BESTÄLLARE

Halmstad Högskola

3 ÖVERSIKTLIG BESKRIVNING AV PROJEKTET

Eleykel-uthyrningssystem, där studenter kan via en hemsida hyra en elcykel som sedan hämtas ut och lämnas tillbaka i ett cykelgarage.

3.1 Syfte och mål

Syftet med projektet är att bidra till en bättre miljö, samt göra en tjänst för campus genom att underlätta för dess studenter att ta sig till olika platser.

Målet med projektet är att få ett fungerande system som är anpassat för Halmstads studenter.

3.2 Leveranser

I januari skall systemet vara färdigt och fungerade. Cykelgarage och elcykel levereras fysiskt till anläggningen där garaget skall stå. Inloggning till server och dokumentation levereras elektroniskt.

3.3 Begränsningar

Vi begränsar oss till att få igång huvudgaraget, samt att vi utgår ifrån att vi bara har en station och inte flera stationer runt om i Halmstad.

4 FASPLAN

Projektet sträcker sig mellan augusti till januari. Nedan beskrivs de tre faser som projektet är uppdelat i.

4.1 Före projektstart

Före projektstart är vi i en fas som kallas förstudie, där vi tillsammans tar fram en planlösning, en kravspecifikation och en testspecifikation. I detta stadiet ingår även förstudier om de olika delarna som ska ingå i projektet, som t.ex. vilka programmeringsspråk som är mest lämpliga att använda inom ett visst område.

4.2 Under projektet

Under projektets gång ska vi utveckla de olika delsystemet efter de krav som ställts i kravspecifikationen. Samt föra dokumentation för de olika delsystemen.

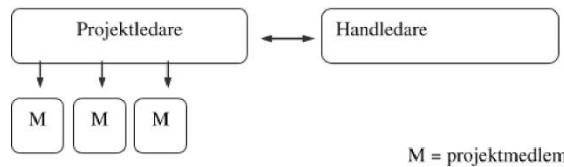


2018-10-17

4.3 Efter projektet

När vi närmar oss slutfasen inom utvecklingsprojektet, skall gruppen hålla en presentation om projektet och vilka resultat gruppen uppnått. Samt ska varje individ i gruppen skriva en rapport som redovisar just de delar individen har var delaktigt i.

5 ORGANISATIONSPLAN FÖR HELE PROJEKTET



Vi har en projektledare som har hand om kommunikation mellan handledare och gruppmedlemmar men har även sitt egna ansvarsområde inom projektet..

Sen har vi projektmedlemmar som har hand om varsitt område i projektet.

Handledaren finns för att bolla idér och ta fram kontakter och även komma med tips inför projektet,

5.1 Villkor för samarbetet inom projektgruppen

Innan projektets start så skrevs ett kontrakt mellan gruppmedlemmarna. Där klargjordes vilka samarbetsregler som gäller och hur vi tänker hantera eventuella brister i samarbetet. Samt att vi redovisade hur kommunikationen och beslutstagandet skulle ske.¹

5.2 Definition av arbetsinnehåll och ansvar

Johannes Hellgren	projektledare (mek)
Göran Trivic	server(dat)
Niklas Eriksson	garage & cykel (elt)
André Kopfinger	garage & cykel (elt)

¹ [Gruppkontakt](#)



2018-10-17

6 DOKUMENTPLAN

Dokument	Ansvarig/ godkänd ansvarig	Syfte	Distribueras till	Färdigdatum
Kravspec.	Hans-Erik Eldemark	Definierar alla krav på systemet	Kursansvarig	2018-10-04
Testspec.	Hans-Erik Eldemark	Definierar all tester på kraven som skall utföras	Kursansvarig	2018-10-20
Tidsplanning	Hans-Erik Eldemark	Hur lång tid varje uppgift skall ta	Kursansvarig	2018-10-05
Elschema	Projektgrupp	Få reda på hur allt är kopplat	Projektgrupp	2018-10-30
Gruppkontrakt	Projektgrupp	Kontrakt för uppförande säkerhetsställs	Projektgrupp	2018-09-05

7 UTBILDNINGSPLAN

Utbildning som skall göras i projektet, både som i grupp och individuellt är inplanerade i tidsplanen.²

7.1 Egen utbildning

Utbildning av programmering i BeagleBoard med JavaScript och server-konstruktion med WordPress, SQL och PHP. Utbildning i versionshantering-verktygen Git och GitKraken, samt hantering av verkstadsmiljöer. Utbildning av Henrik Johansson i konstruktionen av garaget.

8 RAPPORTERINGSPLAN

Var och en i gruppen kommer redovisa varsin tidrapport där man skriver in det man gjort under projektet och hur lång tid det har tagit. Det kan vara t.ex utbildningar, research, föreläsningar och eget arbete. Vi kommer också skriva varsin slutrapport om projektet, där det kommer stå t.ex hur det har gått i projektet och vilka lärdomar vi har tagit del av.

Vi kommer också hålla några muntliga redovisningar, en halvtidsredovisning där vi ska redovisa vår kravspecifikation samt kursplan. Vi ska också hålla en slutlig presentation där vi redovisar vår bakgrundstanke, vision, arbete och slutlig produkt.

²

<https://docs.google.com/spreadsheets/d/1s9jBM30U6sBFtxR0C-52mz1VRJx-yuJ92zskeNUxU0/edit#gid=1681842348>



2018-10-17

9 MÖTESPLAN

Vi har möte med vår handledare Hans-Erik varje onsdag där vi går igenom vad vi har gjort samt om det har kommit några nyheter om garaget. På gruppmöten går vi igenom vad varje person har gjort och uppdaterar de andra. Vi dokumenterar våra gruppmöten på Google Drive.

10 RESURSPLAN

Nedan beskrivs vilka som arbetar med projektet, vilka material, lokaler vi har tillgång till samt vilken budget vi har.

10.1 Personer

Johannes Hellgren, Göran Trivic, Niklas Eriksson ,André Kopfinger ,Heltid.

Hans-Erik Eldemark ,handledare möte en gång i veckan , kan få kontakt med via mail.

Henrik Johansson , jobbar på AES Nordic AB. Jobbat med konstruktionen av garaget. Finns till förfogande 2 timmar för genomgång av hur garaget har fungerat och för att svara på frågor.

Wagner Ourique De Moraes, jobbar på högskolan. Kontakt via mail eller möte i skolan.
Ska hålla en kort genomgång om kommunikation mellan server/databas och hemsida.

10.2 Material

Projektet kräver att vi har tillgång till garagemodulen samt OffCourse elcyklar. I FabLab har vi tillgång till verktyg och mätinstrument.

10.3 Lokaler

Vi har fått tillgång till en verkstad som ligger i Fab Lab. Lokalen kan användas under dagtid och vi ska få tillgång via vårt student ID.

10.4 Ekonomi

Vi har en budget på 3000 kr att handla för.

11 MILSTOLPAR OCH BESLUTSPUNKTER

Innehåller vår milstolpar och beslutspunkter.

11.1 Milstolpar

Nr	Beskrivning	Datum
1.	Kravspecifikationen och projektplanen är klar	2018-10-04
2.	Garaget på plats i Fab Lab	2018-09-28
3.	Sätta på ström till garaget	2018-10-20



2018-10-17

4.	Automatisk laddning av cykel	2018-10-30
5.	Binda knappsats med student-ID	2018-11-30
6.	Få igång kommunikation mellan server och garage	2018-12-05

11.2 Beslutspunkter

Nr	Beskrivning	Datum
0	Godkännande av projektdirektiv, beslut att starta förstudie	2018-09-06
1	Godkännande av kravspecifikation, beslut att starta förberedelsefasen	2018-10-04
2	Godkännande av projektplanering, beslut att starta utförandefasen	2018-10-11

12 AKTIVITETER

Nr	Aktivitet	Beskrivning	Beräknad tid tim
1.	Granskning	Granskning av liknande projekt samt mjukvara	5
2.	Kravspecifikation	Lista de krav vi har för projektet	20
3.	Projektplan	Skriva planen vi har för projektet	10
4.	Utbildning	Utbildning	40
5.	Dokumentering	Dokumentering	20
6.	Rapport	Skriva slutrapport.	40
7.	Möten	Gruppmöten och handledarmöten	60
8.	Presentationer	Förbereda och utföra presentationer	40
9.	Tester	Utföra Tester	40

13 TIDPLAN

Se bifogad tidplan för förväntade aktiviteter, veckoplanering för aktiviteter och förväntad arbetstid.³

³

<https://docs.google.com/spreadsheets/d/1s9jBM30U6sBFBtxR0C-52mz1VRJx-yuJ92zskeNUxU0/edit#gid=1681842348>



2018-10-17

14 FÖRÄNDRINGSPLAN

När man stöter på något problem får detta diskuteras och sedan får man komma fram till någon lösning. Om man inte skulle komma på någon lösning får handledaren informeras.

Vid förseningar så får man komma fram till vad som skall prioriteras, om man inte kommer fram till något får handledaren informeras.

15 KVALITETSPLAN

En CE-märkning betyder att produkten överensstämmer med grundläggande krav på exempelvis hälsa, säkerhet, funktion, miljö. Samt att föreskriven kontrollprocedur har följs. Vår produkt skulle behöva en CE-märkning om den skulle säljas inom EU. För vår produkt ställer EU krav på de olika delarna såsom elektrisk utrustning⁴, byggprodukt⁵ och hiss⁶. Andra viktiga märkningar för vår produkt skulle kunna vara kvalitetssystem ISO 9001 och informationssäkerhet ISO 27001.

Fördelarna med ISO 9001 för vår organisation hade varit att organisationen blir mer organiserad och effektiv samt att det stärker varumärket. Att följa ISO 14001 gör så att produkten hela tiden utvecklas på ett miljöeffektivt sätt, med mindre spill och effektivare energianvändning.

Konkurrenskraften skulle öka då produkten blir kostnadseffektiv och miljöeffektiv. ISO märkningen skulle också ge ökat förtroende för vårt företag och produkt och därfor ge större konkurrenskraft gentemot andra produkter som inte följer ISO standarden.

15.1 Granskningar

Granskning av kravspecifikation och projektplan kommer att utföras av handledaren. Elschema och dokumentation av delsystemen kommer kontrolleras av gruppmedlemmar samt projektledaren. Det kommer ske en slutgiltig granskning av examinator.

15.2 Testplan

Alla funktionella krav ska testas under projektets gång och vi har satt upp milstolpar för när olika mål ska nås och vara klara. Se tidplan och testfalls dokument.⁷

16 RISKANALYS

Se bifogad riskanalys⁸ för vår diskussion kring risker som hotar utgången av projektet, samt våra åtgärder för att minimera riskerna.

17 PRIORITERINGAR

Vid förseningar är de viktigt att vi prioriterar leveransdatum då detta är en prototyp vilket innebär att mycket kommer ändras fram tills produkten ska ut i marknaden.

⁴ <https://eur-lex.europa.eu/legal-content/SV/TXT/HTML/?uri=CELEX:32014L0035&from=EN>

⁵ <https://eur-lex.europa.eu/legal-content/SV/TXT/HTML/?uri=CELEX:32014L0033&from=EN>

⁶ <https://eur-lex.europa.eu/legal-content/SV/TXT/HTML/?uri=CELEX:32011R0305&from=EN>

⁷ <https://docs.google.com/document/d/1VBCdIA7A3tUPouJMXZGzbLrG8iQhOIG23WwaTiVSns/edit>

⁸ <https://docs.google.com/document/d/1HY0mXqI8Is-fLDT32Jla8VJ715qcq5UEXZkvphodqN0/edit>



2018-10-17

Så kvalite får mindre prioritering då man kan finslipa kvaliteten ifall prototypen blir godkänd. Kostnaderna har även hög prioritet då budgeten bara ligger på 3000kr vilket innebär ifall förseningarna har som konsekvens att kostnaderna kommer öka måste budgeten diskuteras på nytt så snabbt de går.

18 PROJEKTAVSLUT

Projektet avslutas med en presentation av arbetet inför kursansvarig och andra projektgrupper. Samt att individuella rapporter skall lämnas in till kursansvarig. Projektet skall även visas upp på utställning inför årskurserna under. All material och dokumentering ska lämnas till Högskolan i Halmstad. Ifall de blir möjligt att fortsätta med projektet som examensarbete kan deltagarna välja detta och utveckla sina färdigheter inom projektarbetet.

19 REFERENSER

Elektroniska källor

EUROPAPARLAMENTETS OCH RÄDETS DIREKTIV 2014/35/EU av den 26 februari 2014 om harmonisering av medlemsstaternas lagstiftning om tillhandahållande på marknaden av elektrisk utrustning

EUROPAPARLAMENTETS OCH RÄDETS DIREKTIV 2014/33/EU av den 26 februari 2014 om harmonisering av medlemsstaternas lagstiftning om hissar och säkerhetskomponenter till hissar

EUROPAPARLAMENTETS OCH RÄDETS FÖRORDNING (EU) nr 305/2011 av den 9 mars 2011 om fastställande av harmoniserade villkor för saluföring av byggprodukter och om upphävande av rådets direktiv 89/106/EG

Opublicerade källor

Tidsredovisning - Tidsplan för projektet

<https://docs.google.com/spreadsheets/d/1s9jBM30U6sBFtxR0C-52mz1VRJx-yuJ92zskeNUxU0/edit#gid=1681842348>

Testfall - Testfall för alla funktionella krav

<https://docs.google.com/document/d/1VBCdIA7A3fUPouJMXZGzbLrG8iQhOIG23WwaTiiVSns/edit#heading=h.mes6picv8x8s>

Riskanalys - Diskussion av risker och åtgärder

<https://docs.google.com/document/d/1HY0mXql8Is-fLDT32Jla8VJ715qcq5UEXZkvpbodqN0/edit#heading=h.gjdgx8>

7.3 Tidsrapportering

Tidsredovisning																							
		Student: Johannes Hellgren																					
AKTIVITETER		TID	TIDPLAN (när), veckonummer																				
Nr	Beskrivning av arbetet	timmar	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	1	2	3	
1	Föreläsning	3,5	3,5	3,5	3,5	3,5	3,5	1,7					1,7										
2	Handledarmöte	1	1	1	1	0,5	0,5	1	1	1	1	1	1	0,5	0,5	1							
3																							
4	Förstudie	2																					
5	Gruppkontrakt	4																					
6	Server förstudie																						
7	Kravspecifikation		3	4	3	2	1									3							
8	Projektplan				2	1																	
9	Riskåtgärder					1																	
10	Testfall				2																		
11	Utbildning för verkstad																						
12	GitHub tutorial/setup														2								
13																							
14	Genomförande																						
15	<i>Server</i>																						
16	Databas - MySQL - phpMyAdmin							6	4						5								
17	Hemsida - Wordpress							4	8						8	3							
18	RPi 3 - ModbusTCP Client till/från databas php skript																						
19	RPi 3 - ModbusUDP Client																				10	13	
20																							
21	Gateway							2	12	5	6	4											
22	BeagleBoard																						
23	Arduino Uno - ModbusRTU Master														3	10	15	19			9	5	
24	Arduino Uno - ModbusTCP Server																			10	3	4	
25	Arduino Uno - ModbusUDP Server																				9	10	
26																							
27	<i>Garage</i>																						
28	Com-Module - ModbusRTU Slave															5	12	8	15				
29	Com-Module - Styrning av kretskort																			8	11	13	
30																							
31	Övrigt														3								
32	Möte med Henrik (AES)															2	2	1	7	4	8	10	
33	Individuell rapport														3								
34	Uppgift - Hållbar produktutveckling															4							
35	Uppgift - Etik																						
36	Halvtidspresentation														2								
37	Examination förberedelse																			18	24		
38																							
39																							
40																							
41																							
42																							
43																							
44																							
45																							
46																							
47																							
48																							
49																							
50																							
51																							
52																							
53																							
54																							
55																							
56																							
57																							
58																							
59																							
60																							
61																							
62																							
63																							
64																							
65																							
66																							
67																							
68																							
69																							

7.4 Server : Databas

```
1  ?>php
2
3  require_once dirname(__FILE__) . '/../../Phplib/ModbusMaster.php';
4  $servername = "localhost";
5  $username = "root";
6  $password = "elcykel";
7  $wordpress = "wordpress";
8  $conn = new mysqli($servername, $username, $password, $wordpress);
9
10 // Create Modbus object
11 $modbus = new ModbusMaster("192.168.1.177", "TCP");
12
13 try {
14     // FC 1
15     $recData = $modbus-&gt;readCoils(1, 62, 1);
16     echo"$recData";
17 }
18 catch (Exception $e) {
19     // Print error information if any
20     echo $modbus;
21     echo $e;
22     exit;
23 }
24 if($conn-&gt;connect_error){
25     die("Connection failed: " . $conn-&gt;connect_error);
26 }
27 while(TRUE){
28     $checkexists = "SELECT * FROM Orders";
29     if($conn-&gt;query($checkexists)){
30         $checkorder = "SELECT userID,active,cykelID FROM Orders WHERE active = TRUE";
31         $result = $conn-&gt;query($checkorder);
32         if($result-&gt;num_rows&gt;0){
33             while ($row = $result-&gt;fetch_array()) {
34                 $userid = $row["userID"];
35                 $ac = $row["active"];
36                 $cykelid = $row["cykelID"];
37                 $flag = TRUE;
38                 echo $cykelid;
39             }
40         }else{
41             echo "can't reach ";
42         }
43         $result-&gt;close();
44 }</pre
```

```
if($ac == TRUE && $flag == TRUE){
    $result = "UPDATE Cykel_Fack SET available = FALSE WHERE cykelID = $cykelid";
    if($conn->query($result)){
        echo " cykelid är inte längre tillgänglig";
    }
} else{
    echo " cykelid hittas inte";
}
$userid->close();
$ac->close();
$result->close();
$flag->close();
}

if($recData == TRUE){
    $updaterafack = "UPDATE Cykel_Fack SET available = TRUE WHERE cykelID = 2";
    if($conn->query($updaterafack)){
        echo"fack uppdaterad";
    }else{
        echo"kunde inte uppdatera fack";
    }
    $updateraorder = "UPDATE Orders SET active = FALSE WHERE cykelID = 2 AND userID = $userid";
    if($conn->query($updateraorder)){
        echo"order är nu arkiverad";
    }else{
        echo"order kunde inte arkiveras";
    }
}
$cykelid->close();
}

// Print read data
echo "</br>Data:</br>";
var_dump($recData);
echo "</br>";
$conn->close();
```

7.5 Server : UDP

```
echo "UDP Client Startup \n";

//Send/receive UDP messages to/from this IP and port
// Arduino Uno
$server_ip = '192.168.1.177';
$server_port = 8888;
// My PC
//$$server_ip = "192.168.1.1";
//$$server_port = 51258;

$beat_period = 1;

error_reporting(~E_WARNING);

// Create a UDP socket
// AF_INET: IP4 address
if (!$socket = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP)) {
    $errorcode = socket_last_error();
    $errmsg = socket_strerror($errorcode);
    die("Couldn't create socket : [$errorcode] $errmsg \n");
}
echo "Socket created \n";

// Bind the source address
if (!socket_bind($socket, "192.168.1.10", "8888")) {
    $errorcode = socket_last_error();
    $errmsg = socket_strerror($errorcode);
    die("Could not bind socket : [$errorcode] $errmsg \n");
}
echo "Socket bind OK \n";
```

```

socket_close($socket);

function readCoils($id, $address, $number) {
    global $server_ip, $server_port, $socket;
    $message = chr($id) . chr(0x01) . chr(0x00) . chr($address) . chr(0x00) . chr($number);

    print "Sending readCoils \n";
    socket_sendto($socket, $message, strlen($message), 0, $server_ip, $server_port);
    print "Time: " . date("%r") . "\n";

    print "Received response from: \n";
    $r = socket_recvfrom($socket, $buf, 512, 0, $server_ip, $server_port);
    $array = bin2hex($buf);
    echo "$server_ip : $server_port -- " . $array . "\n";

    return $array;
}

function writeSingleCoil($id, $address, $value) {
    global $server_ip, $server_port, $socket;
    $message = chr($id) . chr(0x05) . chr(0x00) . chr($address) . chr(0x00) . chr($value);

    print "Sending writeSingleCoil \n";
    socket_sendto($socket, $message, strlen($message), 0, $server_ip, $server_port);
    print "Time: " . date("%r") . "\n";

    print "Received response from: \n";
    $r = socket_recvfrom($socket, $buf, 512, 0, $server_ip, $server_port);
    $array = bin2hex($buf);
    echo "$server_ip : $server_port -- " . $array . "\n";

    return $array;
}

```

7.5 Com-Module : Modbus

```

143 void RTUPoll() {
144     if (!RS485.available()) {
145         return;
146     }
147
148     // Store message in buffer
149     j = -1;
150     while (RS485.available()) {
151         j++;
152         packetSize++;
153         packetBuffer[j] = RS485.read();
154         delay(10);
155     }
156     printBuffer();
157
158     // Read message
159     j = 0;
160     while (packetBuffer[j] != 0x01) {
161         j++;
162         if (packetBuffer[j] == NULL) {
163             Serial.println("SOH not found");
164             clearBuffer();
165             return;
166         }
167     }

```

```
168 if (packetBuffer[j+1] != SLAVE_ID) {
169     Serial.println("Message not for me");
170     clearBuffer();
171     return;
172 }
173
174 // Send response depending on function code
175 switch (packetBuffer[j+2]) {
176     case 0x01:                      // Read coil status
177         ReadCoilStatus();
178         break;
179     case 0x05:                      // Write single coil
180         ForceSingleCoil();
181         break;
182     default:                         // Function code not implemented
183         // if nothing else matches, do the default
184         Serial.print("Error, Function code: ");
185         Serial.print(packetBuffer[2], HEX);
186         Serial.println(" is not implemented");
187         clearBuffer();
188         return;
189     break;
190 }
191
192 clearBuffer();
193 }

195 void ReadCoilStatus() {
196     byte addressHi = packetBuffer[j+3];
197     byte addressLo = packetBuffer[j+4];
198     byte numberHi =  packetBuffer[j+5];
199     byte numberLo =  packetBuffer[j+6];
200
201     // Check if coil address is valid
202     int coilValue = ModbusRTUServer.coilRead(addressLo);
203     if (coilValue != 0 && coilValue != 1) {
204         Serial.print("Error, Coil address: ");
205         Serial.print(addressLo, HEX);
206         Serial.println(" not found");
207         clearBuffer();
208         return;
209     }
210     // Check if number of coils requested is valid
211     if (numberLo <= 0 || numberLo > numCoils) {
212         Serial.print("Error, Number of coils: ");
213         Serial.print(numberLo, HEX);
214         Serial.println(" is invalid");
215         clearBuffer();
216         return;
217 }
```

```
219 // Read coil values
220 byte data[numberLo];
221 for (int i=0; i<numberLo; i++) {
222     data[i] = ModbusRTUServer.coilRead(addressLo + i);
223 }
224
225 if (packetBuffer[j+7] != 0x04) {
226     Serial.println("EOT not found");
227     clearBuffer();
228     return;
229 }
230
231 Serial.println("Sending response");
232 // Respond to master with coil values
233 preTransmission();
234
235 RS485.write(SOH);
236 RS485.write(SLAVE_ID);
237 RS485.write(ReadCoilStatusCode);
238 RS485.write(byte(sizeof(data)));
239 for (int i=0; i<sizeof(data); i++) {
240     RS485.write(data[i]);
241 }
242 RS485.write(EOT);
243
244 postTransmission();
245 }
```

```
247 void ForceSingleCoil() {
248     byte addressHi = packetBuffer[j+3];
249     byte addressLo = packetBuffer[j+4];
250     byte dataHi =  packetBuffer[j+5];
251     byte dataLo =  packetBuffer[j+6];
252
253     // Check if coil address is valid
254     int coilValue = ModbusRTUServer.coilRead(addressLo);
255     if (coilValue != 0 && coilValue != 1) {
256         Serial.print("Error, Coil address: ");
257         Serial.print(addressLo, HEX);
258         Serial.println(" not found");
259         clearBuffer();
260         return;
261     }
262
263     // set the value of the coil
264     if (dataLo == 0x01) {
265         Serial.println("Set to HIGH");
266         ModbusRTUServer.coilWrite(addressLo, 1);
267     }
268     else if (dataLo == 0x00) {
269         Serial.println("Set to LOW");
270         ModbusRTUServer.coilWrite(addressLo, 0);
271     }
```

```
272     else {
273         Serial.print("Error, Force Single Coil data value: ");
274         Serial.print(dataLo, HEX);
275         Serial.println(" is invalid");
276         clearBuffer();
277         return;
278     }
279
280     if (packetBuffer[j+7] != 0x04) {
281         Serial.println("EOT not found");
282         clearBuffer();
283         return;
284     }
285
286     // Respond to master to confirm data write to coil
287     Serial.println("Sending response");
288     preTransmission();
289     RS485.write(SOH);
290     RS485.write(SLAVE_ID);
291     RS485.write(ForceSingleCoilCode);
292     RS485.write(addressHi);
293     RS485.write(addressLo);
294     RS485.write(dataHi);
295     RS485.write(dataLo);
296     RS485.write(EOT);
297     postTransmission();
298 }
```