CODEN:LUTEDX/(TEIE-3046)/1-70/(2014))

Industrial Electrical Engineering and Automation

En snabb prototyp av ett M2M- system

Sam Jabbar

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

En snabb prototyp av M2M lösning

LUNDS UNIVERSITET
Campus Helsingborg

LTH Ingenjörshögskolan vid Campus Helsingborg
Avdelning för Industriell Elektroteknik och Automation

Examensarbete:
Sam Jabbar

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Lunds universitet
Lund 2014

Sammanfattning

I rapporten beskrivs det hur en snabb prototyp av M2M-system byggs. Prototypen består av en hårdvara(.NET Gadgeteer) och en klient som kommunicerar med hårdvaran via en server som är skriven på node.js plattform. Servern tar emot och sänder tillbaka data vilket kommer vara kommunikationsbiten mellan klienten och hårdvaran.[1]

Arbetet inleds med förstudier om hur varje program och plattform fungerar samt hur de ska användas tillsammans i hårdvaran för att kunna bygga ett fungerande system.
Det innefattar också att samla information om hur emulatorn skall användas och att skriva kod för att få en uppfattning om Gadgeteer, .NET Micro Framework, C#, JSON, Telnet, node.js och Rest API.

Det har skrivits två olika typer av servrar och en websida (klient) för att växla service och module strängar mellan Gadgeteer och websidan .

• Net server som tar emot TCP sockets.
• Socket.io server som tar emot Web Sockets.

Det har skapats två olika klasser/ metoder för att koppla gadgeteer till servern.

• TCP metod för att sätta upp en TCP / IP kommunikation med node.js server.
• Socket.io är en färdig klass för att sätta upp kommunikation med node.js server.

•Det har också skrivits en klient sida som används för att skapa strängar som skickas sedan vidare till servern via databasen. Den används också för att ta bort strängar.

Nyckelord: Examensarbete, Rapport, Gadgeteer, C#, .NET MF, Node.js, JSON.Netmf, TCP/IP, Socket.io, Rest API, Mongo DB.

Abstract

The report describes how the quick prototyping of M2M-systems are built. The prototype consists of hardware (.NET Gadgeteer) and a client that communicates with the hardware through a server that is written in node.js platform. The Server receives and sends back data which will be the communication part between the client and the hardware.[1]Instead of having a client that sends data to the server it was chosen to let the server read data from a file.

Work begins with prestudies on how each application and platform works and how to use them together in the hardware to build a working system.
It also includes gathering information on how the emulator works and to write codes to get an idea of the Gadgeteer, the .NET Micro Framework, C #, JSON, Telnet, Rest API and node.js.

There have been written two different types of servers to send and receive data between Gadgeteer and the client:
• Net server that receives TCP sockets.
• Socket.io server that receives Web Sockets.

There have been two different classes / methods to connect to the server Gadgeteer:
• TCP method to set up a TCP / IP communication with node.js server.
• Socket.io completes class that sets up a communication with node.js server.

• A client side was written and it is used to build a Json string that will be sending to the server via a database. It can also be used to delete a Json string.

Keywords: Thesis, Reports, Gadgeteer, C #, .NET MF, Node.js, JSON.Netmf, TCP / IP, Socket.io, Rest API, Mongo DB.

Förord

Denna rapport är en del i examensarbetet som utgör den avslutande delen i min utbildning till Högskoleingenjör Elektro- med Automationsteknik på Lunds Universitet Campus Helsingborg, LTH. Arbetet motsvarar 22.5 högskolepoäng och har utförts under år 2014 tillsammans med Data Ductus AB.

Speciellt tack till Mats Lilja, min examinator på LTH och Göran Edin handledare på Data Ductus AB. Examensarbetet hos Data Ductus AB har varit väldigt spännande, lärorikt och intressant.

Helsingborg/Lund den :

Sam Jabbar

Innehållsförteckning

Ordlista

API: Application Programming Interface

CE: Embedded Compact

CIL: Common Intermediate Language

CLR: Common Language Runtime

HAL: Hardware Abstraction Layer

HTTP: Hypertext Transfer Protocol

IDE: Integrated Development Environment

IOT: Internet of Things

IP: Internet Protocol

JSON: JavaScript Object Notation

M2M: Machine to Machine

MMU: Memory management unit

NPM: Node Packaged Manager

PAL: Platforms Abstraction Layer

REST: Representational State Transfer

SDK: Software Development Kit

TCP: Transmission Control Protocol

VB: Visual Basic

XML: Extensible Mark-up Language

SSL: Secure Sockets Layer

CSS: Cascading Style Sheets

HTML: HyperText Mark-up Language

1 Inledning

1.1 Bakgrund

Internet of things (IoT) är ett projekt som startades 1999 och omfattar koppling av enkla samt avancerade elektroniska apparater till nätverket som till exempel smartphones och persondatorer. IoT uppskattas år 2020 utgöra 100 miljarder enheter. Analytiker anser att etablerade plattformar på nätet kommer att kompletteras av enklare inbäddade enheter i en växande maskin-till-maskin kommunikation. Dessa inbäddade enheterna kommunicerar kontinuerligt med varandra för att underlätta det dagliga livet. Vanliga redskap som till exempel tvättmaskiner och väckarklockor kan tillämpas i M2M-kommunikationen.[1]

Idag är det tidskrävande att bygga ett M2M-system. Det tar lång tid att gå från idé till produktion av hårdvara och mjukvara för att testa en idé.

M2M systemet i det här examenarbetet består av själva hårdvaran som kopplas till olika moduler, en klient som kommunicerar med hårdvaran via servern, en server som tar emot och sänder tillbaka data som kommer vara kommunikationsbiten mellan klienten och hårdvaran.

En enklare metod för att förkorta produktionstiden av hårdvara är att använda en färdig hårdvara som heter Gadgeteer där det är väldigt enkelt att ansluta GPS, RFID, pekskärmar, temperatursensorer etc.

Gadgeteer programmeras med hjälp av C# eller VB och servern skrivs i node.js medan klienten kan vara en applikation i mobilen eller en websida.

1.2 Syfte och målsättning

Syftet med examenarbetet är att bygga ett M2M-system där systemet automatiskt ska kunna detektera vilka moduler som är kopplade till hårdvaran och vilka modulkombinationer som fungerar ihop , till exempel en kamera och en sensor som kopplas till Main board ska fungera som ett larm. Kameran börjar ta bilder när sensorn detekterar en rörelse.

Detta M2M-system byggs med hjälp av Gadgeteer Main board som är hårdvaran och med hjälp av node.js skrivs en server som tar emot och skickar tillbaka data som beskrivs av JSON strängar. Kommunikationen mellan Gadgeteer och servern sker via Websockets eller TCP/IP-protocollet. En klient som är skriven i HTML och JavaScript kopplas till servern via MongoDB databasen.

1.3 Begränsning

Begränsningar som görs i examensarbetet är att arbetet görs enbart för Gadgeteer , servern skrivs i node.js, kommunikationen mellan Gadgeteer och servern ska implementeras direkt med TCP/IP eller alternativt utnyttja WebSocket som protokoll. Servern kopplas till en klient och kommunikationen mellan servern och klienten ska ske med hjälp av REST API.

1.4 Problemformulering

- **Förstå Gadgeteer och hur det konfigureras samt skriva kod på ett sådant system.**
  Gadgeteer upptäcker moduler som stödjes av den och två valfria moduler väljs. Funktionerna i Gadgeteer sätts igång lokalt och uträttas vanliga uppgifter t.ex. en kamera ska kunna ta bilder. Gadgeteer skickar händelser till servern.

- **Skriv en serversida.**
  Servern skrivs i Node.js och kommunikation med Gadgeteer sker via Websockets eller TCP/IP Protokoll.

- **Skriv en klientsida.**
  En klientsida skrivas. Den ska kunna kommunicerar med servern. Klienten skrivs i javaScript och kommunikationen sker via REST API.

1.5 Kort om DataDuctus

Data Ductus AB grundades år 1989 och idag är drygt 130 anställda fördelade på olika kontor i Skellefteå, Luleå, Uppsala, Stockholm och Malmö samt Longmont, CO i USA.[2] Data Ductus arbetar med allt från stora, internationella flerårsprojekt till små lokala insatser. Det handlar om allt från analys- och strategiarbeten till utveckling, drift och övervakning. Data Ductus har lång erfarenhet inom området "Maskin-till-maskin" (M2M), vilket syftar på tekniker som tillåter enheter att kommunicera med andra

enheter eller ett centralt system. Ett exempel på detta är en sensor som används för att fånga en händelse så som temperatur eller nivå, då ändringen förmedlas via ett nätverk (trådlöst, kabel, eller hybrid). Mottagen data skickas vidare till ett program (mjukvara) som översätter det till meningsfull information.[3]

2 Metod

Arbetet inleds med förstudier om hur varje program och plattform fungerar samt hur de ska användas tillsammans för att kunna bygga ett fungerande system.
Det innefattar också att samla information om hur emulatorn skall användas och att skriva kod för att få en uppfattning om Gadgeteer, .NET Micro Framework, C#, JSON, Telnet, node.js, Mongo DB och REST API.

Efter att det bildats en uppfattning om hur delarna skall fungera ihop delas projektet in i mindre delar enligt problemformuleringen ovan.

Litteratursökningen fortsätter under programmeringen för att kunna lösa problemen som uppstår under tiden. Den ursprungliga idéen med att modulerna i Gadgeteer inte skall vara förprogrammerade i Main board utan skickas information genom en driver användes inte i detta arbete eftersom det visade sig att modulerna inte var tillräckligt smarta för detta (det tas upp i senare del av arbetet). Innan hårdvaran blev tillgänglig behövdes serverkoden testas. Detta gjordes via Telnet för att kommunicera med servern.

2.1 Källkritik

Jag har använt mig av bloggar som källor eftersom personerna som skrev i bloggarna har skrivit originellt material d.v.s. de har kommit på egna exempel eller själva bloggen är källan. Eftersom Gadgeteer använder sig av .NET Micro Framework som är öppen källkod var det svårt att hitta trovärdiga källor. Den mesta informationen som hittades kommer från olika bloggar och forum.

3 Teknisk bakgrund

Den här delen av rapporten tar upp alla de nödvändiga program och plattformar som behövs för att bygga upp M2M-systemet. Den tekniska bakgrunden delas upp i sex huvudrubriker som beskriver hela M2M-systemet:

- **Hårdvara**
- **Server**
- **Kommunikationen mellan Hårdvaran och servern**
- **Klient**
- **Kommunikationen mellan servern och Klienten**
- **Databas**

Under varje huvudrubrik finns det underrubriker som tar upp tillbehören till varje del.

3.1 Hårdvara

3.1.1 Microsoft .NET Gadgeteer

Gadgeteer har en design som förenklar applikationsutveckling så mycket som möjligt. Gadgeteer bygger på en lödfri metod som låter utvecklaren bygga en enhet enkelt och snabbt som går lätt att bygga om igen. Moderkortet kan anslutas till olika moduler med jackbara kablar, till exempel en kamera eller en knapp. Detta ger en stor frihet i att utforma både användning och form hos hårdvaran.[4]

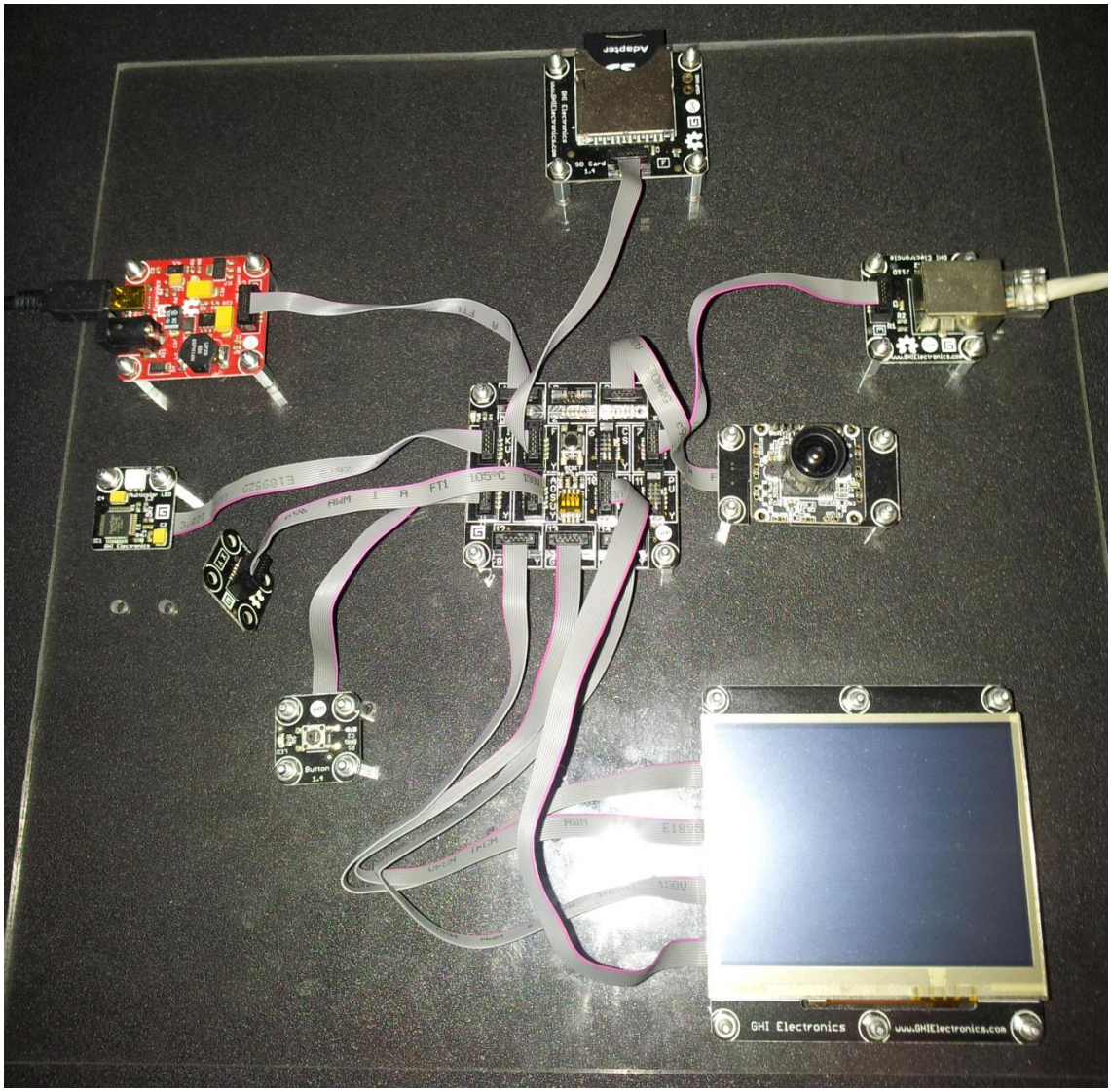


Fig 1: FEZ Spider kopplad till olika moduler

För att programmera enhetens funktionalitet används Visual studio express eller pro och programmeras i C# eller Visual Basic.

Gadgeteer är byggd kring en objektorienterad och händelsebaserad metod. Detta innebär att varje fysisk hårdvarumodul representeras i koden av motsvarande programvaruobjekt med vissa egenskaper som utför relevanta metoder och hanterar specifika händelser. En programvaruhändelse genereras till exempel automatiskt när en knapp trycks ned och detta utlöser i sin tur ett kodblock som kallas en händelsehanterare som säger till programmet att köra.[1]

Ett exempel på hur enkelt och lätt det blir att programmera genom användningen av

händelsebaserad metod:

```
void ProgramStarted()
      {
       // Initialize event handlers here.
 button.ButtonPressed += new Button.ButtonEventHandler(button_ButtonPressed);

       // Do one-time tasks here
          Debug.Print("Program Started");
      }
        // waiting for a button press.
    void button_ButtonPressed(Button sender, Button.ButtonState state)
      {
          Debug.Print("Button pressed.");
       }
```

Att Gadgeteer använder sig av händelsebaserade metoder förenklar skapandet av många program och hjälper utvecklaren att bekanta sig med händelsebaserad programmering på stationära och mobila plattformar.
Ett mjukvarubibliotek kapslar in varje fysisk moduls funktionalitet genom en snabb hög-nivå API. Den höga abstraktionsnivån gör det lätt för utvecklaren att programmera moduler.[1]

Detta sänker inlärningströskeln för nybörjare som ska utveckla enheter, men begränsar inte den flexibilitet som finns tillgänglig för mer erfarna utvecklare. Den plattform som ligger till grund för Gadgeteer är .NET Micro Framework som är en öppen källkod med en öppen programvarustack, som innehåller omfattande bestämmelser för nätverk. Gadgeteers nätverksAPI bygger på öppen källkod på ett sätt som stödjer ett kompakt, enkel designmönster som hanterar REST-full webbförfrågningar med text, bilder eller byte-strömmar. Varje Gadgeteer-modul kommer med drivrutiner, som finns samlade i en SDK-fil som kallas Gadgeteer SDK. En XML-beskrivning av socketens typ och en bild ingår också. SDK-filen kan laddas ned kostnadsfritt från internet.[5]

En Visual grafikdesigner som är integrerad i Visual Studio IDE efter installation av SDK och .NET Gadgeteer Core, kan automatiskt ansluta modulerna med relevanta moderkortsuttag. Detta leder till att eventuella konflikter kan upptäckas innan modulerna kopplas till Main board.[6]

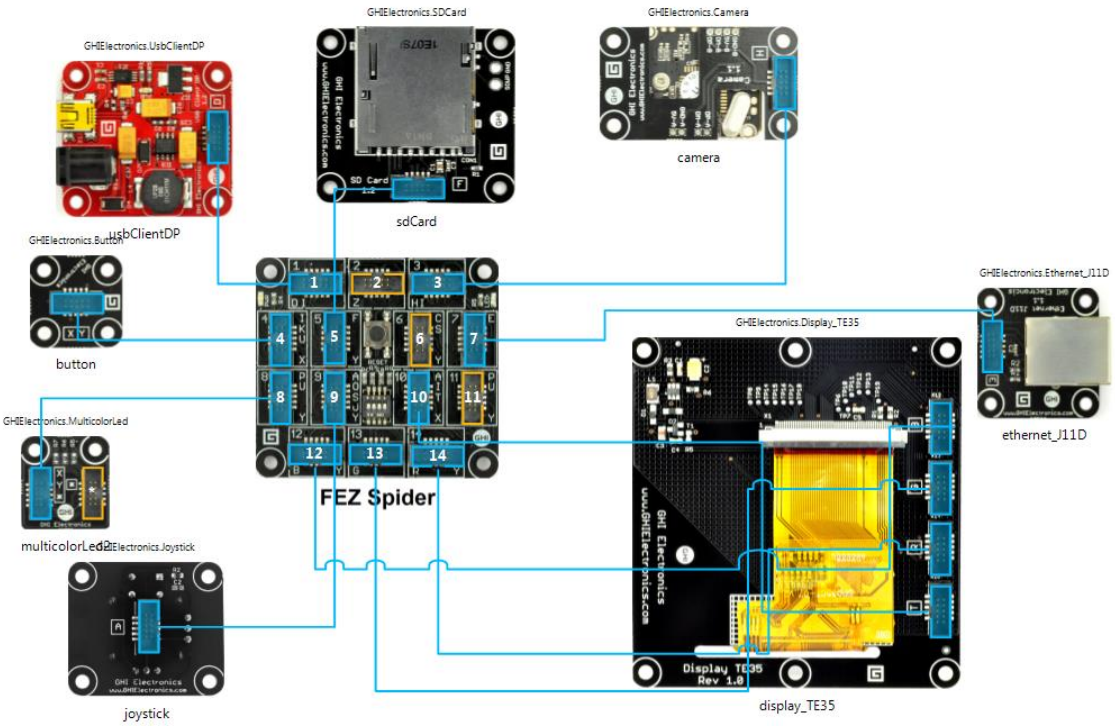


Fig 2: Visual Studio visar hur FEZ Spider ska kopplas till olika moduler

3.1.2 .NET Micro Framework

.NET Micro Framework, .NET Compact Framework och .NET Framework är tre liknande ramverk, men de har olika storlekar för att kunna användas i olika miljöer.[7]

.NET Framework körs på datorer och inte på mindre enheter, eftersom det är ett mycket stort ramverk. Dessutom har hela ramverket många bibliotek som inte är användbara på mindre enheter.[7]

Av den anledningen skapades .NET Compact Framework. I Compact Framework har onödiga bibliotek tagits bort vilket möjliggjort en förminskning av ramverket. Denna mindre version körs på Windows CE och smarta telefoner. .NET Compact Framework

är mindre än .NET Framework, men det är fortfarande för stort för att köras i minienheter eftersom det kräver ett operativsystem.[7]

.NET Micro Framework är den minsta versionen av dessa ramverk. Flera bibliotek har uteslutits och det blev därför OS-oberoende. .NET Micro Framework ger också en utbyggbar hårdvaruemulator för snabba prototyper och felsökning. .NET Micro Framework kräver inget underliggande operativsystem. En förminskad version av Common Language Runtime (TinyCLR) sitter direkt på hårdvaran, varför ramverket oftast kallas en startbar Runtime. Runtime-modulen använder endast några hundra kilobyte av RAM-minnet och det kräver inte att processorn har en minneshanterings enhet (MMU). Därför kan .NET Micro Framework köras på små och billiga 32 - bitars processorer vilket innebär att strömförbrukningen hålls nere.[7]

3.1.2.1 TinyCLR

I CLR exekveras mellanliggande språkkod och följande centrala tjänster tillhandahålls: Hardware Abstraction Layer (HAL), Plattforms Abstraction Layer (PAL) och NET Micro Framework CLR. HAL är kopplad till hårdvaran. HAL har tillgång till hårdvara och kringutrustning genom att tillhandhålla funktioner. HAL använder sig av underliggande operativsystem för att ge funktionalitet när ett operativsystem körs. PAL ger ytterligare abstraktioner av HAL, som till exempel timer och minnesblock. CLR består av exekveringsmotor och garbage collector. [7]

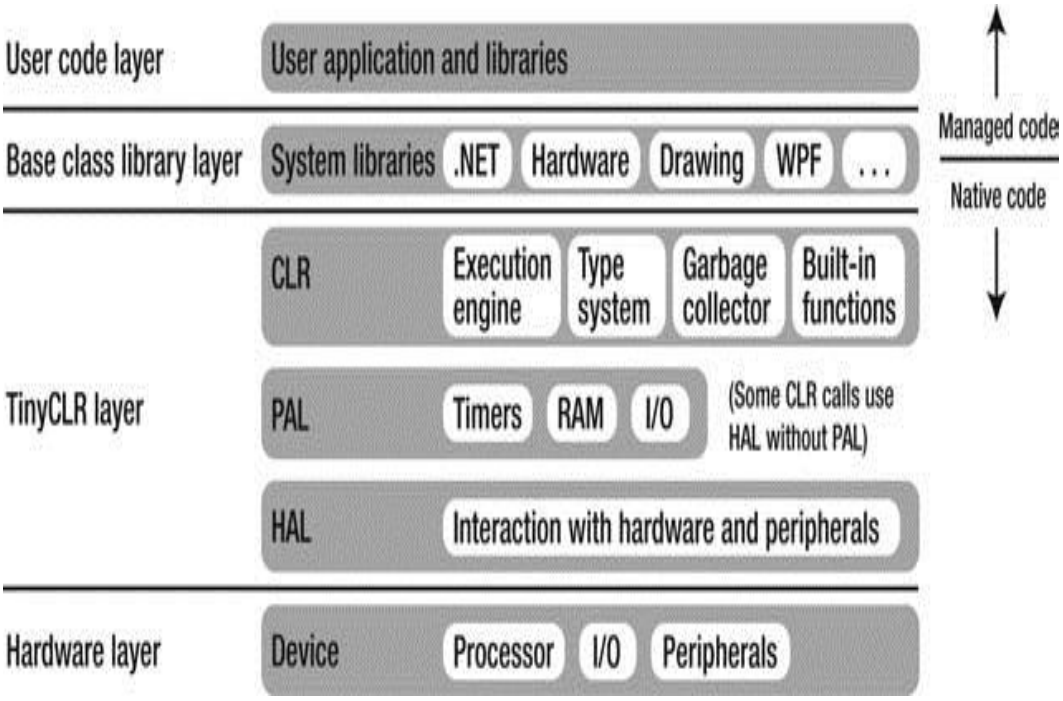


Fig 3: Arkitektur av .net micro framework.[7]

3.1.2.2 TinyCLR uppgifter

Kod som körs och hanteras av CLR kallas "managed code", medan kod som inte riktar CLR kallas opåverkad "native code".
Den CLR utför mellanliggande språk kod och tillhandahåller följande centrala tjänster och förmåner:

- Automatisk minneshantering med hjälp av garbage collector.
- Ämne hantering och synkronisering.
- Undantagshantering.
- Strikt typ säkerhet.
- Säker och robust hanterad kod.
- Avlusa tjänster.

CLR använder sig av en garbage collector som automatiskt rensar oanvända minnesblock och hanterar trådar genom att ge tidsluckor till de enskilda trådar och metoder för att synkronisera tillgång till fördelade resurser. Managed koden exekveras under kontroll av CLR, som tar hand om referenser till objekt, så det är inte nödvändigt att hantera pekare som är används i nativ programkod. Med Managed kod, går det inte

att komma åt minnesblock när de inte längre tillgängliga eftersom CLR frigör objekt endast när de inte refereras längre. [7]

3.1.3 SDK (Software Development Kit)

En Software Development Kit är ett paket som används av utvecklare för att underlätta arbetet eftersom den innehåller en färdigskriven kod och därmed krävs inte att skriva program från grunden. SDK används för att skapa applikationer för ett visst programpaket, t.ex. Framework programvara, hårdvaruplattform, datorsystem, spelkonsol, operativsystem, eller liknande utvecklingsplattform. [8] eller kan vara något så enkelt som att genomföra ett eller flera programmeringsgränssnitt(API: er) i form av en del av bibliotek anslutas till ett visst programmeringsspråk eller att inkludera avancerad hårdvara som kan kommunicera med ett visst inbyggt system.[9] Oftast kan SDK laddas ner direkt via Internet. Många SDK tillhandahålls gratis för att uppmuntra utvecklare att använda systemet eller språket.[10]

3.1.4 Emulator

Microsoft .NET Micro Framework innehåller en kraftfull hårdvaru-emulator som utvecklaren kan förbättra och utöka för att anpassa efter sina behov när denne vill skapa inbyggda system för hårdvara.[10]

3.1.4.1 Emulator Översikt

.NET Micro Framework innehåller en uppsättning verktyg som gör att utvecklaren kan emulera sin hårdvara i mjukvara. Med denna utbyggbara hårdvaru-emulator, kan utvecklaren ha med både sin maskin och programvaran i utvecklingen samtidigt.[11]

När emulatorn skapas, kommer först emulatorkomponenter att byggas och sedan skrivs en XML-konfigurationsfil som innehåller information för att köra konfigurationen, lastning, och initieras.Windows-formulär som ser ut som en LCD, används för att implementera ett användargränssnitt för emulatorprogrammet.[11]

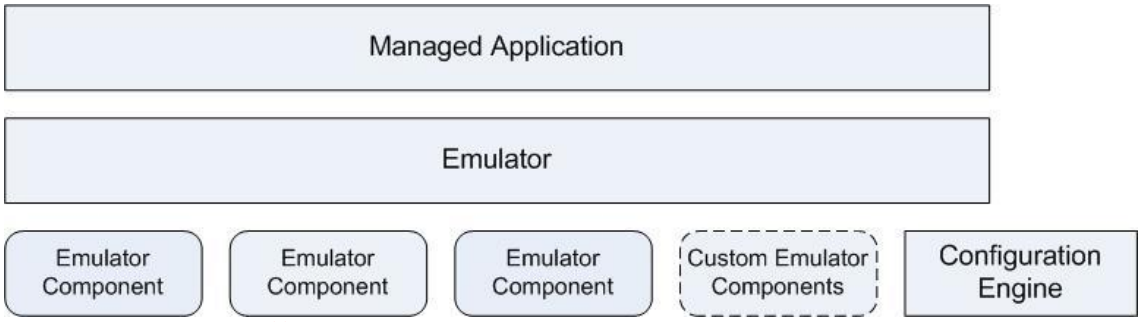


Fig 4: Emulator Arkitektur.[11]

3.1.5 C#

C och C + + är de mest populära programmeringsspråk. C # är en uppdaterad och moderniserad version av C och C + +.
Det innehåller allt du kan förvänta dig av ett modernt språk, som garbage collector och Runtime-validering. Det är också objekt orienterat vilket gör programmen mer portabel och lättare att felsöka och port. Även om C # lägger till en del nya regler för programmering för att minska risken för fel, innehåller det ändå de flesta av de kraftfulla funktionerna som ingår i C/C++. [12]

3.1.5.1 Jämförelse mellan C# och C++

Visual C# Referensimplementation finns endast i Windows Visual Studio Express 2013

- Inga globala variabler eller funktioner
- Pekare kan bara användas i kod block märkta med nyckelordet unsafe
- Säkrare (starkare) typning
- Garbage collector används
- Kompilering till CIL (Common Intermediate Language)
- Exekvering på CLR (Common Language Runtime) (ett slags virtuell maskin).[13]

Foreach och{set; get} var nya språkelement för mig annars är resten ganska likt Java och C++.

3.1.5.2 Foreach

Exempel:

```
int sum;
foreach(int x in nums) {
   Console.WriteLine("Value is: " + x);
   sum += x;
}.
```
[14]

3.1.5.3 Systematisering av set och get

Exempel:

```
public class Customer {

    private int m_id = -1;

    public int ID {
       get { return m_id; }
       set { m_id = value;}
    }


    private string m_name = string.Empty;
    public string Name {

       get { return m_name; }
       set { m_name = value;}
    }
}.
```
[14]

3.1.6 NuGet

NuGet är ett fritt, öppet system för. NET källpakethantering som består av några klientverktyg (NuGet kommandoraden och NuGet Visual Studio Extension).[14]

Microsoft har bidragit till utvecklingen av NuGet projektet. NuGet projektet infördes år 2010, sedan dess många organisationer och privat personer började inse att NuGet ger en stor möjlighet att förbättra och automatisera delar av sina utvecklingsprocesser. Oavsett om utvecklaren arbetar med öppen källkod-projekt eller i en företagsmiljö.[15]

NuGets- klientverktyg ger möjlighet att producera och konsumera paket. Nuget.org är centrala paketet förrådet som används av alla konsumenter. När ett paket installeras, kopierar NuGet filer till projektet och gör automatiskt de ändringar som behövs, till exempel att lägga till referenser och ändra på app.config eller web.config. Om det väljas att ta bort biblioteket, tar NuGet bort filer och sätter tillbaka de ändringar som gjordes i projektet så att inget skräp skall finnas kvar.[16]

3.2 Server

3.2.1 Node.js

Node.js är ett event-drivet I/O ramverk byggt på JavaScript-motorn V8. Det är skapat för att kunna skriva skalbara nätverksprogram som exempelvis webbservrar. Programmeringsspråket som används är JavaScript som exekveras på serversidan.
I node.js utförs nästan inga funktioner som direkt blockerar I/O. Detta leder till att Deadlock inte kan uppstå. Det hanteras genom callbacks, för att istället Vänta på ett resultat från I/O kan Node.js exekvera annan kod under tiden. När I/O är färdig anropas callback-referensen och resultatet hanteras.

Server och klientsida kan skrivas antingen i Sublime som är en enkel text editor eller i Visual Studio. Visual Studio Pro är mest lämplig eftersom den stödjer Gagdeteer också då kan det tas nytta av att spara hela projektet i en plats. Att använda Visual studio för att skriva server och klient kod gör det lättare att hitta fel i koden genom användning av Debug. [17]

3.2.2 NPM

NPM (Node Package Manager) är en pakethanterare för Node.js. Den tillåter utvecklaren att skapa, dela och återanvända moduler i Node.js applikationer. Den kan också användas för att dela kompletta Node.js applikationer. Moduler är helt enkelt kod som ingår i bibliotek som kan återanvändas i olika projekt. [18]

3.2.3 TelNet

Telnet är ett gammalt textbaserat windowsprogram som används för att ansluta till en annan dator eller ett program i själva datorn till exempel en server via Internet. Med hjälp av Telnet kan användaren komma åt andra datorer och ha åtkomst till exempel till e-post, databaser och filer.[19]

3.2.3.1 Hur används Telnet?

Telnet används som en klient som anslutas till en server node.js server. Telnet-klienten finns tillgänglig i Windows. Kommandon som används för att sätt igång Telnet och ansluta till en lokal node.js server är:

1. Skriv i kommando raden: ”telnet”.
2. Sedan skriv ”Open localhost portnummer”. [19]

3.3 Kommunikationen mellan Hårdvaran och server

3.3.1 JSON

JSON (JavaScript Object Notation) är ett datautbytesformat . Det är lätt för människor att läsa och skriva och lätt för maskiner att tolka och generera. Det är baserat på JavaScript. JSON är ett textformat som är helt språkoberoende men använder konventioner som är bekanta för programmerare i C-språkfamiljen, inklusive C, C ++ , C #, Java, JavaScript, Perl, Python, och många andra. Dessa egenskaper gör JSON till ett perfekt datautbytesspråk. [20]

3.3.1.1 JSONs struktur

JSON är en samling av namn / värde-par. På olika språk, realiseras detta som ett objekt, rekord, struct, ordbok, hashtabell, nyckellista, eller associativ array. En annan förklaring är att JSON är en ordnad lista med värden. I de flesta språk, realiseras detta som en matris, vektor, lista, eller sekvens.

Dessa är universella datastrukturer. Så gott som alla moderna programmeringsspråk stödjer dem i en eller annan form. Det är logiskt att ett dataformat som är utbytbar med programmeringsspråk också baseras på dessa strukturer. [21]

JSON installeras genom NuGet direct till Visual Studio projekt.

3.3.1.2 Hur formas ett JSON objekt

Ett objekt är en icke ordnad uppsättning av namn / värde-par. Ett objekt börjar med {(vänster parentes) och slutar med} (höger parentes). Varje namn följs av: (kolon) och namnet / värde paren är separerade med, (kommatecken). [21]

Exempel:
Serialisering av en JSON string:

```
Hashtable hashtable = new Hashtable ();
hashtable. Add("firstName ", John);
hashtable. Add("lastName ", Doe);

string json = JsonSerializer.SerializeObject(hashtable);

Debug.Print(json);

// Output: "{\"firstName\":\"John\",\"lastName\":\"Doe\"}";
```

Deserialisering av en JSON string :

```
string json = "{\"firstName\":\"John\",\"lastName\":\"Doe\"}";

Hashtable hashTable = JsonSerializer.DeserializeString(json) as Hashtable;

Debug.Print(hashTable["firstName"] + " " + hashTable["lastName"]);
```

// Output: "John Doe";

3.3.1.3 JSON och jämförelse med XML

XML är ett annat datautbytesspråk som kräver extra bibliotek för att hämta data från Document Object Model (DOM).

JSON beräknas tolka upp till hundra gånger snabbare än XML i moderna webbläsare, men trots sina påståenden om anmärkningsvärd prestation så finns det argument mot JSON och det inkluderar brist på namespace-stöd och brist på kontroll av indata.[22]

3.3.2 TCP

TCP (Transmission Control Protocol) är en standard som definierar hur ett nätverk kan upprätta och upprätthålla konversation mellan en sändare och en mottagare som vill utväxla data. TCP fungerar med Internet Protocol (IP), som definierar hur paket skall skickas och mottagas. TCP och IP är de grundläggande protokoll som definierar Internet. [23]

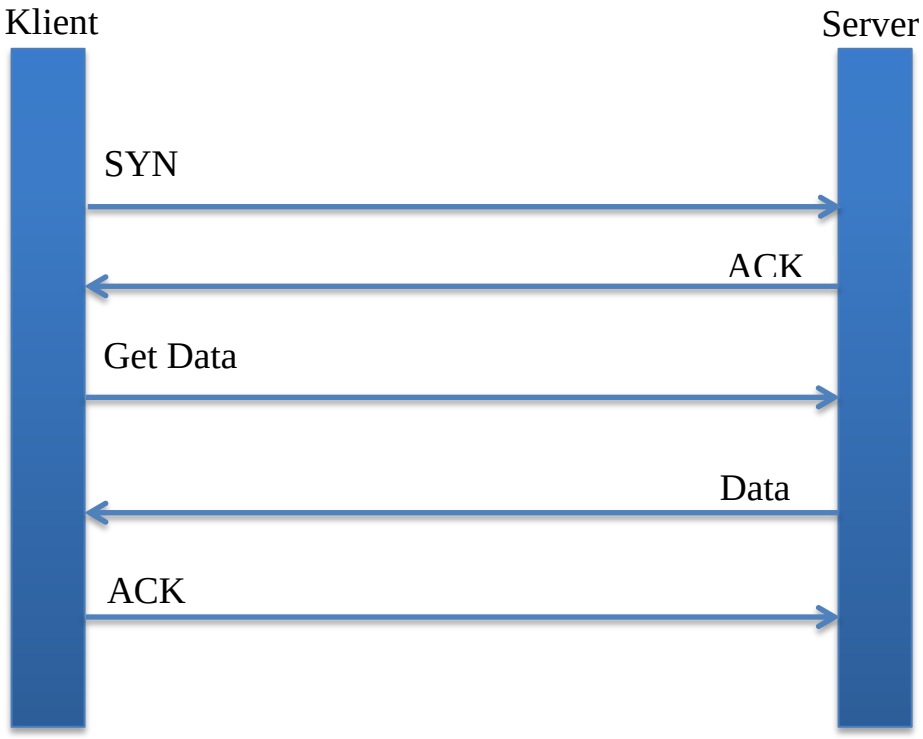


Fig 5: Bilden symboliserar hur kommunikationen mellan en klient och en server upprättas i en TCP/IP kommunikation och hur mottagaren skickar en Ack till sändaren för att informera om att paketet har ankommit.

3.3.3 WebSocket

WebSocket är ett protokoll som ger kommunikationskanaler med full duplex över en TCP-anslutning.[24]

WebSocket är utformad för att implementeras i webbläsare och webbservrar, men det kan användas av vilken klient eller server applikation. Det WebSocket Protokollet är ett oberoende TCP - baserat protokoll.Dess enda relation till HTTP är att det är handskakning tolkas av HTTP-servrar. WebSocket-protokollet möjliggör mer samspelet mellan en webbläsare och en webbplats, underlättar levande innehåll och skapandet av realtidskommunikation. Detta är möjligt genom att tillhandahålla ett standardiserat sätt för servern att skicka innehåll till webbläsaren utan att begäras det av kunden, och gör det möjligt för meddelanden som skickas fram och tillbaka samtidigt hålla förbindelsen

öppen. På detta sätt en tvåvägs (dubbelriktad) pågående kommunikation kan ske mellan en webbläsare och servern.Kommunikationen sker över TCP-port nummer 80, vilket är till nytta för de miljöer som blockerar icke - web Internet-anslutningar med hjälp av en brandvägg. WebSocket protokoll stöds för närvarande i flera webbläsare, inklusive Google Chrome, Internet Explorer, Firefox, Safari och Opera. WebSocket kräver också webbapplikationer på servern för att stödja det. [25]

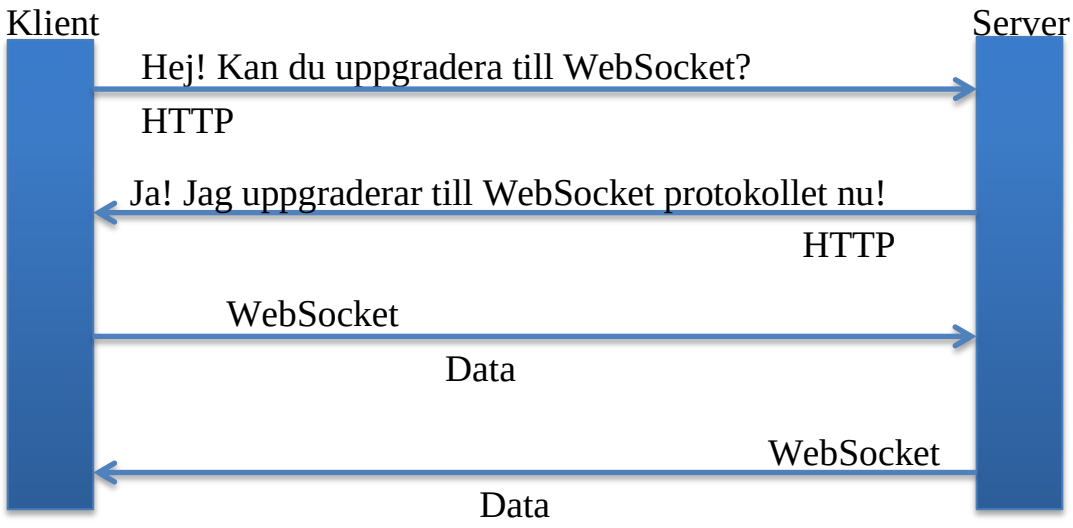


Fig 6: Bilden symboliserar hur kommunikationen mellan en klient och en server upprättas i en Websockets kommunikation . Först uppsätts det en HTTP kommunikation sedan frågar klienten om uppgradering till Websockets.

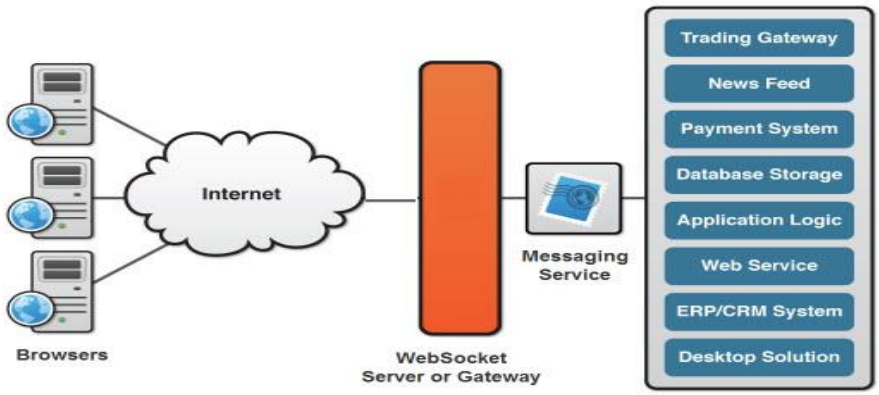


Fig 7: https://www.websocket.org/aboutwebsocket.html.

Websocket gör att fler klienter kan kontakta servern på samma gång och får data den specifika klienten vill ha. Klienten som använder websocket behöver inte vara en webbrowser det kan vara en applikation i en mobil.

3.3.3.1 Varför och när ska Websockets användas?

Förr krävdes det en överanvändning av HTTP för att skapa webbapplikationer med dubbelriktad kommunikation mellan en klient och en server. Detta krävdes för att kunna polla servern för uppdateringar och gjordes samtidigt som HTTP skickade anmärkningar i motsatt riktning för att skilja mellan olika HTTP-anrop.

Detta leder till olika problem. Det första problemet är att servern var tvungen att använda sig av ett antal olika underliggande TCP anslutningar för varje kund. En anslutning för att skicka information till klienten och en ny anslutning för varje inkommande meddelande. Det andra problemet var att varje gång kunden kontaktar servern så skickades det ett meddelande som har en HTTP-header. Wire protokollet har en stor overhead. Tredje problemet är att klientsidans script tvingas att upprätthålla en kartläggning från utgående anslutningar till inkommande anslutning för att spåra svar.

Websocketprotokollet löser problemen genom att använda en enda TCP-anslutning för trafik i båda riktningarna. Websockets tekniken används i en mängd olika webbapplikationer spel, aktiesymboler och fleranvändarapplikationer med samtidig redigering. [25]

3.4 Klient

3.4.1 Bower

Bower är en pakethanterare för webb, den fungerar som NPM som är en pakethanterare för server sidan(Node.js), men istället är den en pakethanterare för webbläsare. Bowers syfte är att hantera front-end (t.ex. websida) tillgångar. Till skillnad från NPM kan Bower ha flera olika sorters filer som till exempel (.js, .css, .html, .png, .ttf). [26]

3.4.2 JavaScript

JavaScript är ett objektorienterad programmeringsspråk för websidor, nästan alla moderna websidor använder sig av javaScript. [27] Den använder typer, operatorer, objekt och metoder precis som andra programmeringsspråk så som C och Java och många andra. Skillnaden mellan JavaScript och Java eller C är att den inte har några klasser istället har den klass funktionalitet mer i objekt prototyper.[28]

3.4.3 HTML

Hypertext Markup Language förkortas till HTML. HTML är en uppsättning av formatkoder eller symboler som lagras i en fil för att sedan läsas av websidan.[29] En annan deffinition är att HTML är ett textualt språk som speciferar hur text och grafik skall visas upp på en websid. HTML kod skrivs i en vanlig text fil och sedan sparas med .htm eller .html på slutet. När en websida besöks så skickas sidan till mottagaren för att den skall visas i motagarnes browser. HTML har speciella taggar som te.x. <title></title> är specialist för att sätta upp en titel till websidan.
<title>Hello World</title>, “Hello World” kommer vara Websidans titel I det här exemplet.[30]

3.4.4 CSS

Cascading Style Sheets, förkortas till CSS. CSS är ett stilmall som används för att beskriva hur ett element som är skriven i HTML eller XML ska presenteras på en skärm, papper eller i tal. [31] CSS mallar används till exempel för att styra teckensnitt, färg, justering av text och objekt mm.[32]

En klient kan vara en applikation (mobil applikation) eller websida som är skriven med hjälp av JavaScript.

Ett snabbt exempel på en javaScript klient som skickar service till Gadgeteer:

Edit This Code: See Result »

```
<!DOCTYPE html>
<html>
<body>

<h1> Send a service to Gadgeteer </h1>

<button type="button"
onclick="document.getElementById('demo1').innerHTML =
'{Data:Service,Service:CAMERA,camera:Camera3,button:Button2}'">
 Camera3+Button2 = Snapchat.</button>
<p id="demo1"+'\n'></p>

<button type="button"
onclick="document.getElementById('demo').innerHTML
='{Data:Service,Service:LED,x:x,button:Button1}'">
 MultiLED+Button1= Disco.</button>

<p id="demo"+'\n'></p>

</body>
</html>
```

Fig 8: http://www.w3schools.com/js/tryit.asp?filename=tryjs_myfirst

Fig 9: http://www.w3schools.com/js/tryit.asp?filename=tryjs_myfirst

Tanken om knappen aktiveras så sänds strängen till servern. Rest Api används för att få data från servern get, delete för att ta bort, put för att ändra, post att skicka data.

3.5 Kommunikationen mellan servern och Klienten

3.5.1 API

API (Application Programming Interface) är en abstraktion som definieras av beskrivningen av ett gränssnitt och beteende gränssnittet.[33] Vid programmering av ett program för att kommunicera med ett separat program eller plattform, kommer du nästan alltid använder ett API för att kunna kommunicera med de andra programmen.[9,34]

3.5.2 Rest API

Representational State Transfer (REST) är ett begrepp som beskriver hur tjänster för maskin till maskin-kommunikation kan tillhandahållas. Varje REST Resurs identifieras av en unik URL och implementerar ett antal standard operationer som tillåter konsumenten att skapa, läsa, uppdatera och radera resurser (CRUD). Dessa operationer representeras av standard HTTP verb som GET, POST, PUT och DELETE. [35]

3.6 Databas

MongoDB är en databas baserad på öppen källkod som är enkel att använda. Här är några enkla exempel på hur man sparar data i MongoDB och hur sökande efter data går till.

```
> db.scores.save({Data: 'Service', Service : 'LED' ,x:'x',button:'Button1'});
WriteResult({ "nInserted" : 1 })
> db.scores.save({Data: 'Service', Service : 'CAMERA' ,camera:'Camera3',button:'Button2'});
WriteResult({ "nInserted" : 1 })
> db.scores.find();
{
    "_id" : ObjectId("54736be21cdcaf7c0c689e42"),
    "x" : "x",
    "button" : "Button1",
    "Data" : "Service",
    "Service" : "LED"
}
{
    "_id" : ObjectId("54736c1240694733307d807a"),
    "camera" : "Camera3",
    "button" : "Button2",
    "Data" : "Service",
    "Service" : "CAMERA"
}
```

Fig 22: MongoDB del 1

http://try.mongodb.org/?_ga=1.159167615.479371167.1415283814

```
> db.scores.find({Service:{ $in: ['CAMERA','LED']}});
{
    "_id" : ObjectId("54737ca440694733307d878c"),
    "x" : "x",
    "button" : "Button1",
    "Data" : "Service",
    "Service" : "LED"
}
{
    "_id" : ObjectId("54737d8c1cdcaf7c0c68a59b"),
    "camera" : "Camera3",
    "button" : "Button2",
    "Data" : "Service",
    "Service" : "CAMERA"
}
```

Fig 23: MongoDB del 2

http://try.mongodb.org/?_ga=1.159167615.479371167.1415283814

$in betyder sök efter Service som heter CAMERA och LED.

```
> db.scores.find({Service:{ $in: ['CAMERA']}});
{
    "_id" : ObjectId("54737d8c1cdcaf7c0c68a59b"),
    "camera" : "Camera3",
    "button" : "Button2",
    "Data" : "Service",
    "Service" : "CAMERA"
}
>
```

Fig 24: MongoDB del 1
http://try.mongodb.org/?_ga=1.159167615.479371167.1415283814

4 Utförande

4.1 Principlösning 1

Upptäckbara moduler

Idéen är att modulerna skall inte vara förprogrammerade i Main board . Tanken var att när modulerna blir kopplade till Main board så ska de presentera sig för Main board och skicka information om sig själva genom att skriva en driver för varje modul som frågar modulen när programmet startar t.ex. vad det är för enhet eller vad den har för ID nummer. Konfiguration klassen rättar sedan till informationen t.ex. att alla kameror från olika företag skall heta ”kamera” så att den nya informationen skickas vidare till servern då servern skall skicka vidare informationen till klienten. Klienten sänder därefter händelser till servern om att installera den koden man behöver för kombinationen av de kopplade modulerna.

4.2 Principlösning 1 fungerar inte!

Anledningen till att idéen inte går att genomföra är att modulerna inte är så smarta så att de kan sända tillbaka information om sig själva. De har inget minne i sig och all information om modulerna finns redan nerladdade i Main board. Detta görs via nedladdning från codeplax hemsida som kallas källkod. källkod innehåller all information om samtliga moduler. Gadgeteer är en öppen källkod och därför kan man teoretiskt ändra på källkoden, men i praktiken är detta en lång process att genomgå för detta examensarbetet och därför valdes det bort.

4.3 Principlösning 2

Ett alternativ till att modulerna skall vara upptäckbara är att använda ett SD-kort som kopplas till Main board från början. Med hjälp av SD-kortet kan modulerna initieras och använda informationen enligt Arkitekturen av Systemet nedan.

4.4 Arkitekturen av M2M-systemet

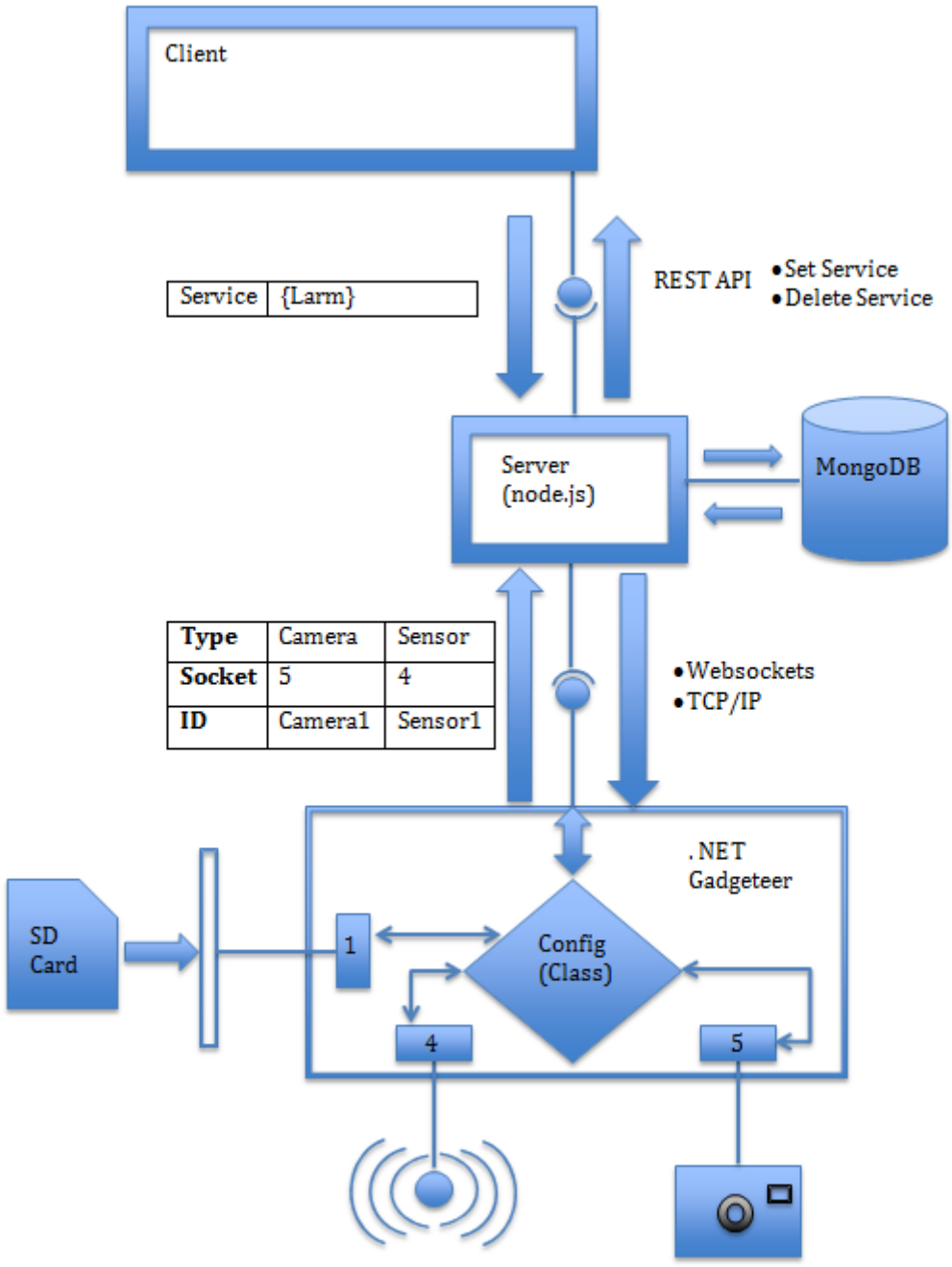


Fig 10: Arkitekturen av M2M-systemet

4.5 Detaljer på Arkitekturen

SD-kortet innehåller information om vilka moduler som skall kopplas till Gadgeteer och vilket ID som varje modul har. Data som kommer in från klienten via servern till Gadgeteer kallas Service det är också en data sträng som kopplar ihop moduler tillsammans.

4.5.1 Exempel

Type	Kamera	Sensor
Socket	12	13
ID	Kamera1	Sensor3

SD-kortet innhåller data om två moduler en kamera och en sensor. Där ”Kamera l” är ID och 12 är socket nummret som modulen ska kopplas till.

Service	Larm	Sensor3,Kamera1

Service stränger kommer från klienten via servern. Den Services namn är Larm och kopplar ihop sensor med ID Sensor3 till Kamera med Id Kamera1. Informationen i SD-kortet är skriven i JSON format för att kunna uppdatera det direkt från servern.

4.5.2 Klasser i M2M-systemet

SD-kortet innehåller filer och varje fil innehåller en JSON sträng som representerar en modul eller en service. Strängarna bildas och Serialiseras i ett separat projekt och skickas det vidare till SD-kortet. SD-projektet har ingen koppling till M2M-systemet den är gjord för att slippa att skriva data i SD-kortet för hand.

Det finns olika metoder och klasser som bygger upp M2M-systemet. Nedan tas det de viktigaste klassarna av M2M-systemet.

4.5.2.1 Init_Ethernet_Connection

I Gadgeteer det går att använda sig av både Ethernet och Wifi moduler. Ethernet modulen användes för M2M-systemet i det här projektet. Ethernet modulen kopplar ihop Gadgeteer till routern och via den kommunicerar den med yttre världen.

Så här initieras ethernet modulen:
GTM.Ethernet_J11D ethernet_J11D = new Ethernet_J11D(7);

4.5.2.2 Use_JSON_Data_From_SDcard_And_Server

Den här klassen har en viktig function för hela systemet. Dess uppgift är att ta emot en JSON string och deserialisera den för att senare både spara informationen genom att anroppa Module_List klassen och använda deserialiserade datan för att sätta igång systemet till exapel att initiera Event Handler för varje module som klassen får data om.

Här är ett exempel från koden:

” button (hashTable ["button"].ToString ()).button.ButtonPressed += new Button.ButtonEventHandler (button_ButtonPressed);”

```
Void button_ButtonPressed ()
{
// do something…
}
```

I den klassen anropas klassenerna Module, Module_lists och All_Modules_Array. Det anropas också metoden Send_To_TCP_Class() som skickar alla sammlade data till servern.

4.5.2.3 Module

Klassen Module innehåller tre modul klasser Button, MulticolorLed, och Camera.
Alla har två viktiga metoder GetID och GetSocket.GetID Har en viktig funktion, den särkylier varje modul från varandra till exepel om vi har två knappar och vi vill att använda de för att de ska göra två olika saker då blir det lättare för mikro processorn att veta viken knapp ska göra vad.GetSocket metoden ger en unikt integer värde för varje module , som pekar på vilken socket nummer på Gadgeteer ska den kopplas till.

4.5.2.4 Module_Lists

Module_Lists klassen innehåller också tre klasser ButtonList, LedList och CameraList.

Alla klasser har en konstruktor som tar emot data och sparar det i en lista. Klasserna har metoder som söker efter en speciell modul med hjälp av Id-värdet. Det som retuneras är Id-värdet och socket nummeret.

4.5.2.5 All_Modules_Array

Klassen All_Modules_Array den sparar alla JSON strängar direkt från Use_JSON_Data_From_SDcard_And_Server klassen för att kunna skicka informationen till klassen Connect_To_Tcp_Server genom metoden Send_To_TCP_Class som finns i Use_JSON_Data_From_SDcard_And_Server.

4.5.2.6 Connect_To_TCP_Server

Den klass sätter kommunikationen mellan Gadgeteer och servern via TCP protokollet. Klassen har två metoder en Send och den andra är Recive. Send metoden innehåller initieringen av protokollet som ser ut på det här sättet:

```
Socket = new Socket (AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);

Socket.Connect (new IPEndPoint (IPAddress.Parse (_host), TCPport));
```

Efter initieringen av Connect metoden så sätts kommunikationen mellan Gadgeteer och server sedan skickas all data som fås av klassen Use_JSON_Data_From_SDcard_And_Server genom metoden Send_To_TCP_Class.

4.5.2.7 My_SocketIO_Client

Klassen ärver några metoder som finns i klassen SocketIO som i sin tur använder SocketIO sig av JDI Websockets protokollet. [36] Klassen SocketIO är en öppen källkod

test projekt som finns på nätet. [37] Metoderna som används är emit och onEvent. Emit används för att skicka data och onEvent för att ta emot data.

4.5.2.8 Klassen Program

Program klassen är Main klass där annropas Init_Ethernet_Connection, My_SocketIO_Client, socketIOClient.connect och metoden ReadfromSD.

4.5.3 Server koder skrivna i node.js

Det skrevs två olika sorters Server kodar, en TCP och en annan socke.io server. Båda tar emot JSON strängar och sparar strängarna i en fil. De kan läsa data från en fil och skickar den vidare till Gadgeteer.

Eftersom det finns två fungerande servrar så användes den ena för att läsa data från en eller fler filar och skicka de vidare till Gadgeteer. Den andra används för att ta emot och spara data i en fil.

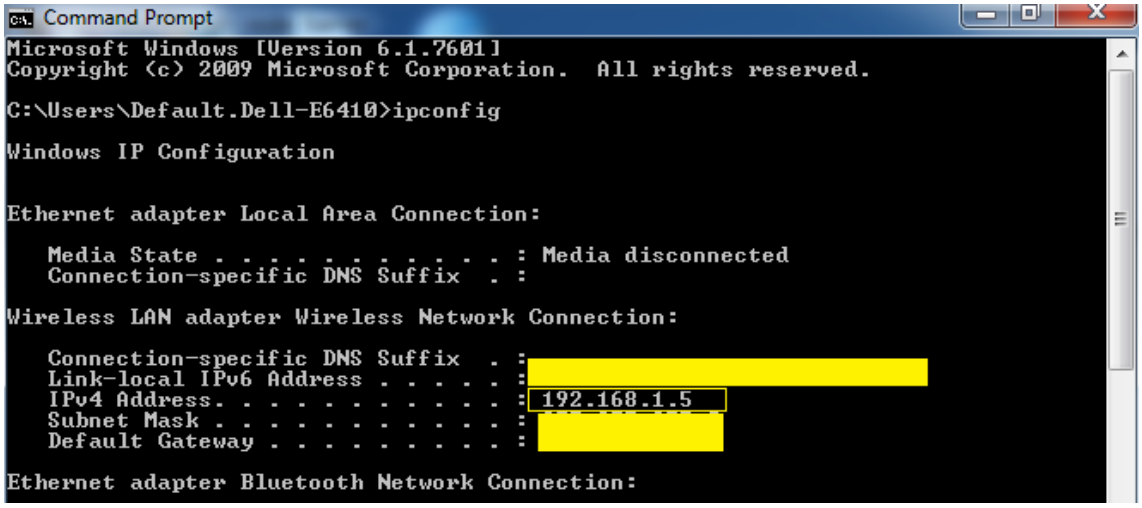


Fig 11: command prompt

För att kunna koppla TCP-serevern till Gadgeteer servern lyssnade på alla IP- addressar som försöker koppla sig till datorn. Gagdeteer anslutar sig till datorns IP-address eftersom där servern finns. Datorns IP-address hittas genom att skriva komman koden ”ipconfig” i Command Pompt som det står på bilden ovan.

4.5.3.1 TCP server

tserver.js log.json

```
1
2   var net = require('net');
3
4   var HOST = '0.0.0.0';
5   var PORT = 443;
6
7   var JSONStream = require('json-stream');
8
9   var filePath=__dirname +'/log.json';
10  var file = require('fs')
11  .createWriteStream(filePath);//,{flags:'a'});
12
13  var es = require('event-stream');
14
15
16
17  var stringifier = es.mapSync(function(data) {
18
19      console.log(' Data on port'+ JSON.stringify(data));
20
21       return  JSON.stringify(data);
22       stringifier.pipe(file);
23     socket.write('Echo server\r\n');
24  });
25
26
27  var server = net.createServer();
28  server.on('connection', function(socket) {
29
30      console.log('Got data on port'+  443 );
31
32    socket.pipe(file);
33    var jsonStream = new JSONStream();
34
35     socket
36      .pipe(jsonStream)
37      .pipe(stringifier, {end: false});
38
39  }).listen(PORT, HOST);
40
41  console.log('Server listening on ' + HOST +':'+ PORT);
42
```

Fig 12:TCP server

En TCP server kännetecknas genom annropet av requier(’net’) klassen. Servern lyssnar på alla IP addressar som försöker kontakta port 443. JSON _stream används för att omvandla alla text strängar till JSON Strängar innan de sparas i filen.

Requier(’fs’) är en file stream som skickar data till log.json. pip(stringifier, {end: false}) tar emot strömmande data som skickas från Gadgeteer och false i koden står gör att Stringifier inte stängas efter att data har tagit slut. Den används för att kunna ta emot data som kommer in hela tiden.

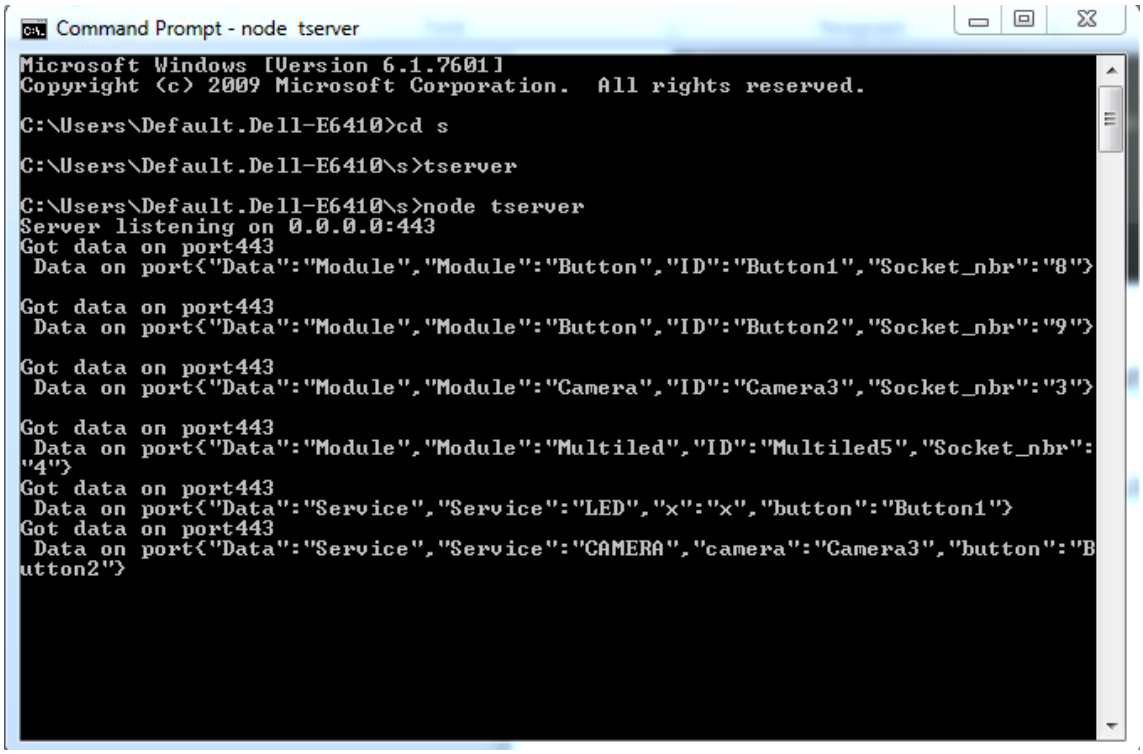


Fig 13: tserver

Servern får data som sparas i en fil som heter Log.json. Det sparade data raderas när det sätts en ny kommunikation. På bilden ovan vias att TCP servern har fått nya data av Gadgeteer och den har fått de via port 443.

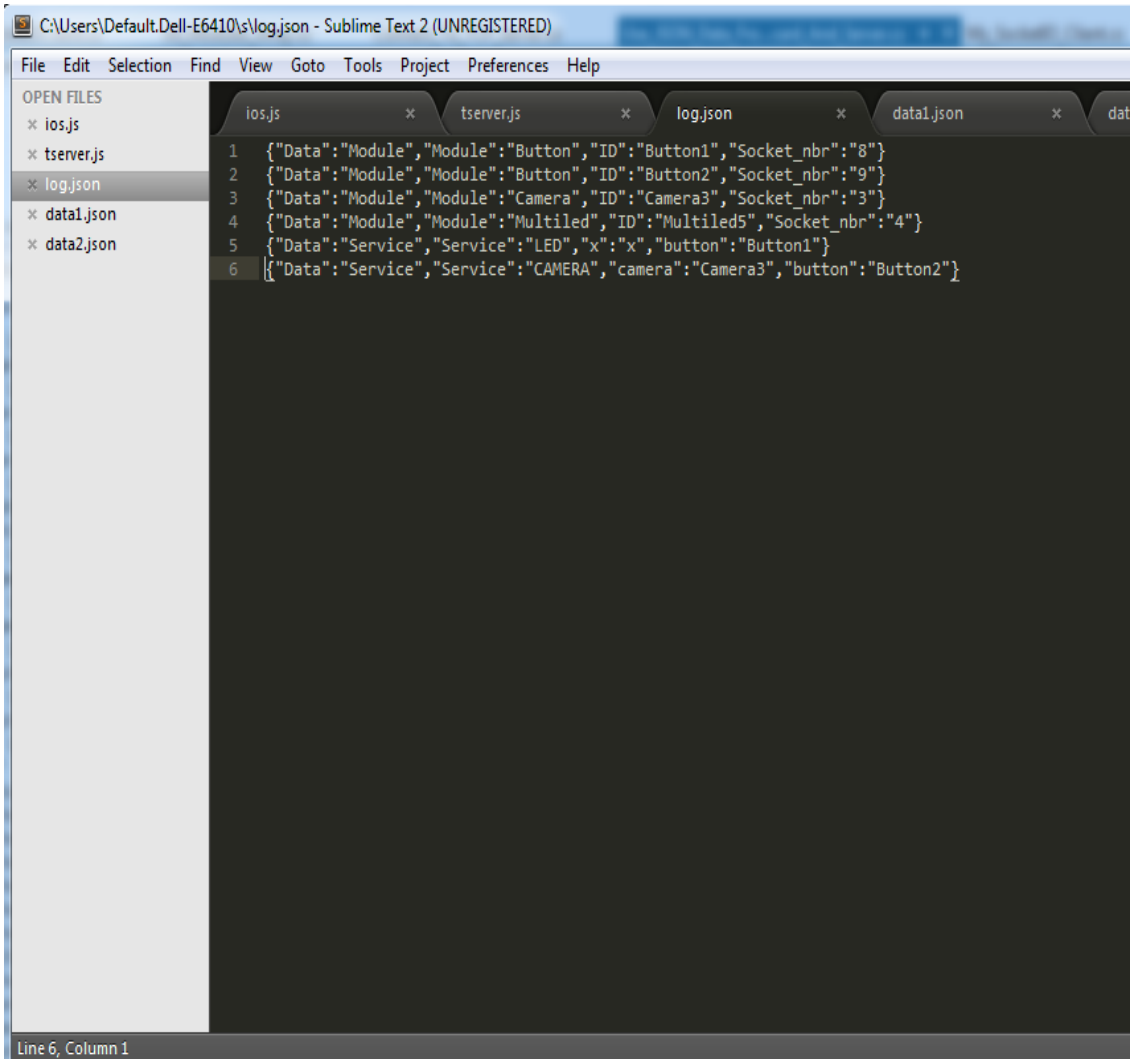


Fig 14: log.json

Så här ser JSON strängarna ut som log.json har fått av Gadgeteer.

4.5.3.2 Socket.io server

ios.js data1.json data2.json

```
var fs = require('fs');

var io = require('socket.io').listen(80);

io.sockets.on( 'connection', function( socket ) {
  console.log( 'connection' );


fs.readFile('./data1.json',function(err, log){


socket.emit('Data'+ (log));

});


fs.readFile('./data2.json',function(err, log){

socket.emit('Data'+ (log));

});

  socket.on( 'login', function( message ) {

    console.log( 'login ' + message );

  });

  });

console.log('server is connected to 80');
```

Fig 15: IO server

Socket.io servern lyssnar på port 80. Kommunikationens sätts när io.socket.on(’connection’) får data som heter connection. Data.json läsas av och informationen skickas via Socket.emit(’Data’+ (log)). Data är namnet på meddelandet och log är strängen.

Data1 innehåller:
'"{Data: Service, Service : LED ,x:x,button:Button1}"'

Data2 innehåller:
'"{Data: Service, Service:CAMERA,camera:Camera3,button:Button2}"'

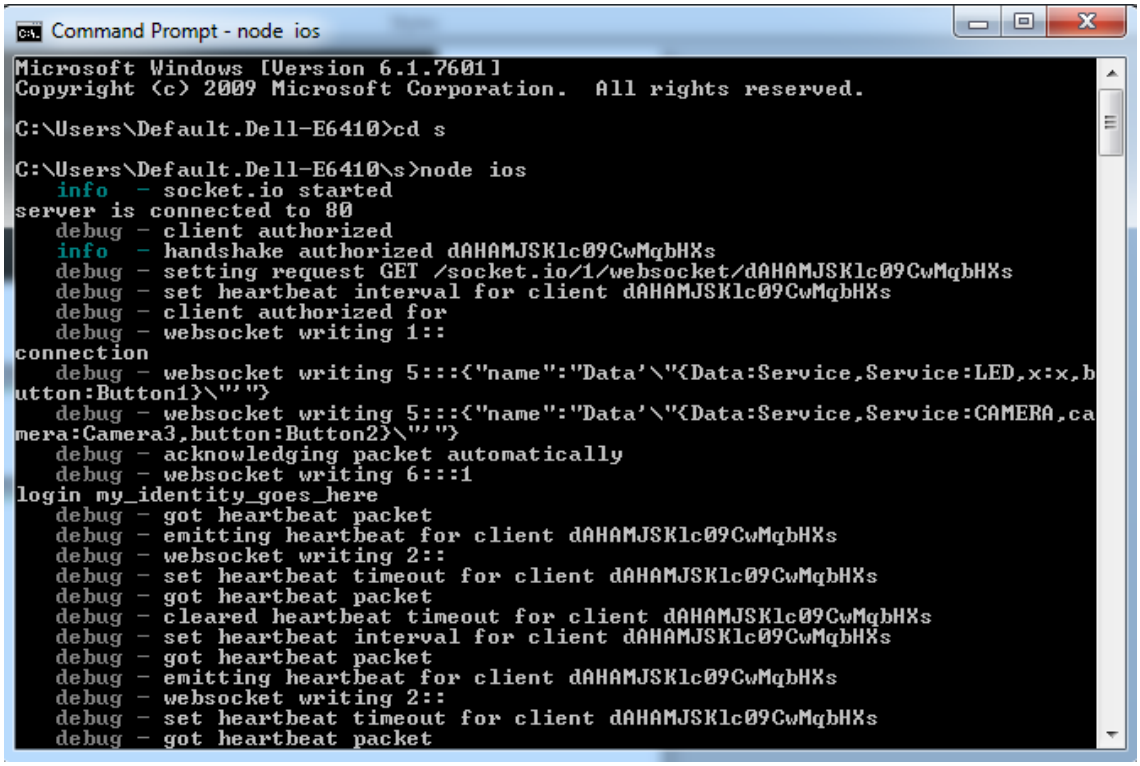


Fig 16: ios.js

4.5.4 Websida skirven i HTML, CSS och JavaScript

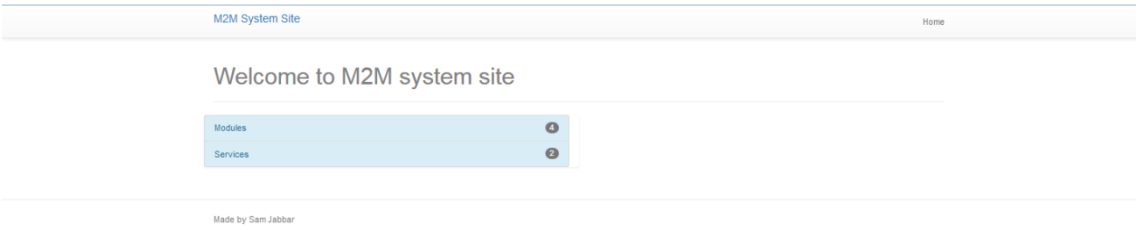


Fig 17: M2M websida Home

Home sidan är den första sidan där det visas att databasen har 4st Module och 2st Service strängar. När Modules knappen trycks kommer man till Modules sidan som ser ut som på bilden nedan.

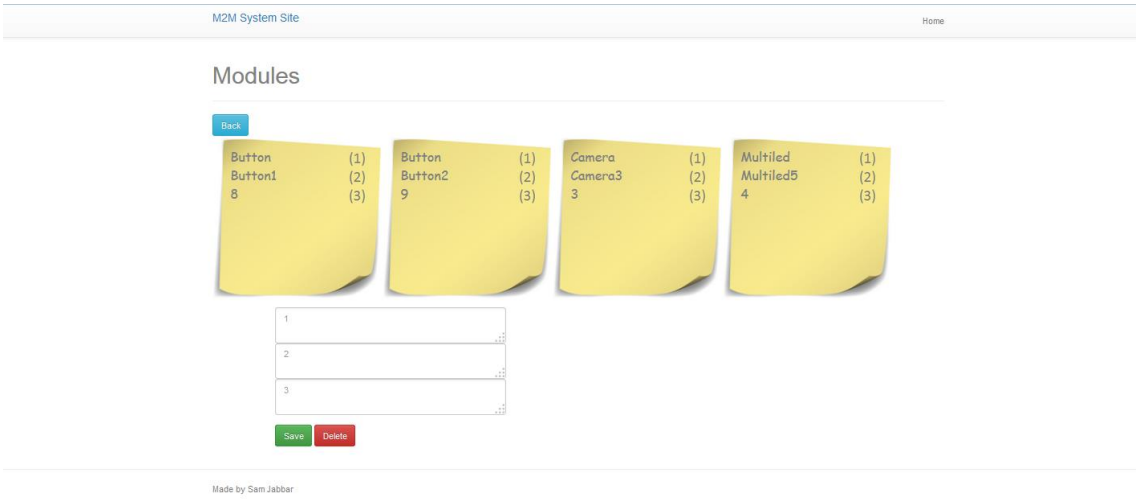


Fig 18: M2M websida Modules

Modules sidan visar vilka module-strängar som finns i databasen. På sidan finns det också tre rutor för att addera eller ta bort en module sträng.

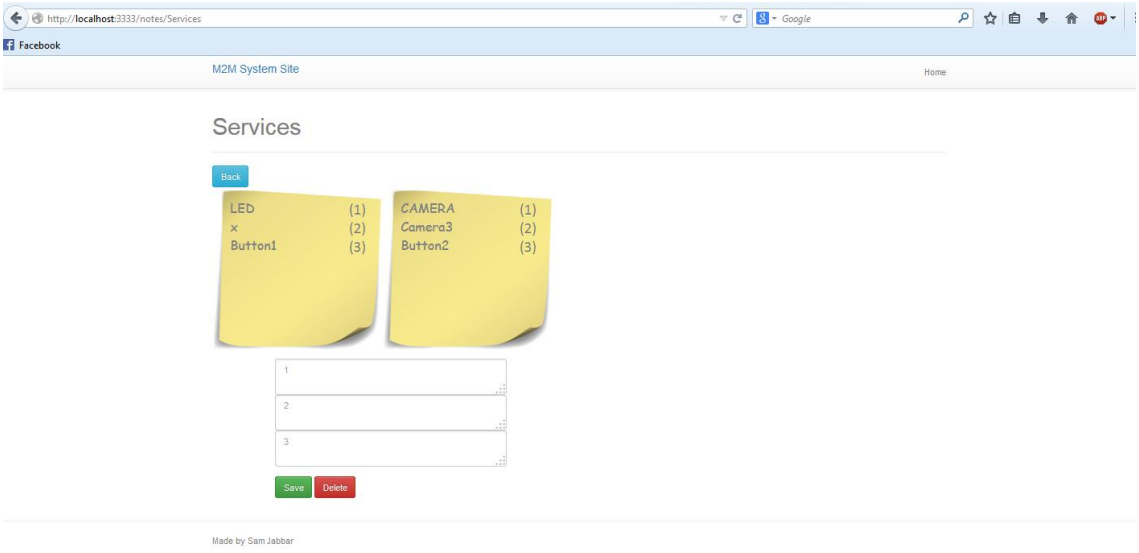


Fig 19: M2M websida Services

Services sidan visar vilka service strängar som finns i databasen. På sidan finns det också tre rutor för att addera eller ta bort en service sträng.

Websidan är skriven med hjälp av bland annat HTML, CSS och JavaScript.

Websidan använder Post, Get och Delete metoder för att kunna skicka, ta emot, och radera API:er.

5 Diskussion och slutsats

Idéen är att modulerna inte skall vara förprogrammerade i Main board . Tanken var att när modulerna blir kopplade till Main board så ska de presentera sig för Main board och skicka information om sig själva genom att skriva en driver för varje modul som frågar modulen när programmet startar.

Som tidigare nämnt så byggdes M2M-systemet med hjälp av Gadgeteer och i node.js skrivs en server som tar emot och skickar tillbaka data som beskrivs av JSON strängar. Kommunikationen mellan Gadgeteer och servern sker via Websockets eller TCP/IP. Där systemet automatiskt vet vilka moduler som är inkopplade till hårdvaran. Detta fungerade dock inte eftersom modulerna inte är tillräckligt smarta så att de kan sända tillbaka information om sig själva. De har inget minne i sig och all information om modulerna finns redan nedladdade i Main board. Detta görs via nedladdning från codeplax hemsida som kallas källkod. Källkod innehåller all information om samtliga moduler. Gadgeteer är en öppen källkod och därför kan man teoretiskt ändra på koden, men i praktiken är detta en lång process att genomgå för detta examensarbetet och därför valdes det bort.

Ett alternativ till att modulerna skall vara upptäckbara är att använda ett SD-kort som kopplas till Main board från början. Med hjälp av SD-kortet kan modulerna initieras och använda informationen enligt Arkitekturen av Systemet. Sedan skall data som finns på SD-kortet uppdateras via nätet då får man ingen nytta av SD-kortet längre men det blir ett alternativ för att ändra på informationen om nätet är avkopplat.

Under projektets gång gick det att få hårdvaran att lyssna på SD-kortet genom att t.ex skriva textfiler som innehåller JSON strängar i SD-kortet. Varje fil innehåller en rad instruktioner antingen att initiera en modul eller att få två moduler eller fler att samarbeta. Detta testades på en kamera, två knappar och en multicolorled vilket fungerade väl.

Det har skrivits två olika sorters servrar med hjälp av Node.js för att ta emot data från Gadgeteer och skicka tillbaka svar:

- Net server som tar emot TCP sockets.
- Socket.io server som tar emot websockets.

Det har skrivits två olika klasser/ metoder skrivna i C# .net Micro Framework för att koppla Gadgeteer till servern:

- TCP metod för att sätta upp en TCP/ IP-kommunikation med node.js server.
- Socket.io färdig klass som sätter upp en kommunikation med node.js server.

Det har också skrivits en klient sida som används för att skapa strängar som skickas sedan vidare till servern via databasen. Den används också för att ta bort strängar.

Strängarna ser ut som på bilden nedan, varje fil innehåller en sträng.

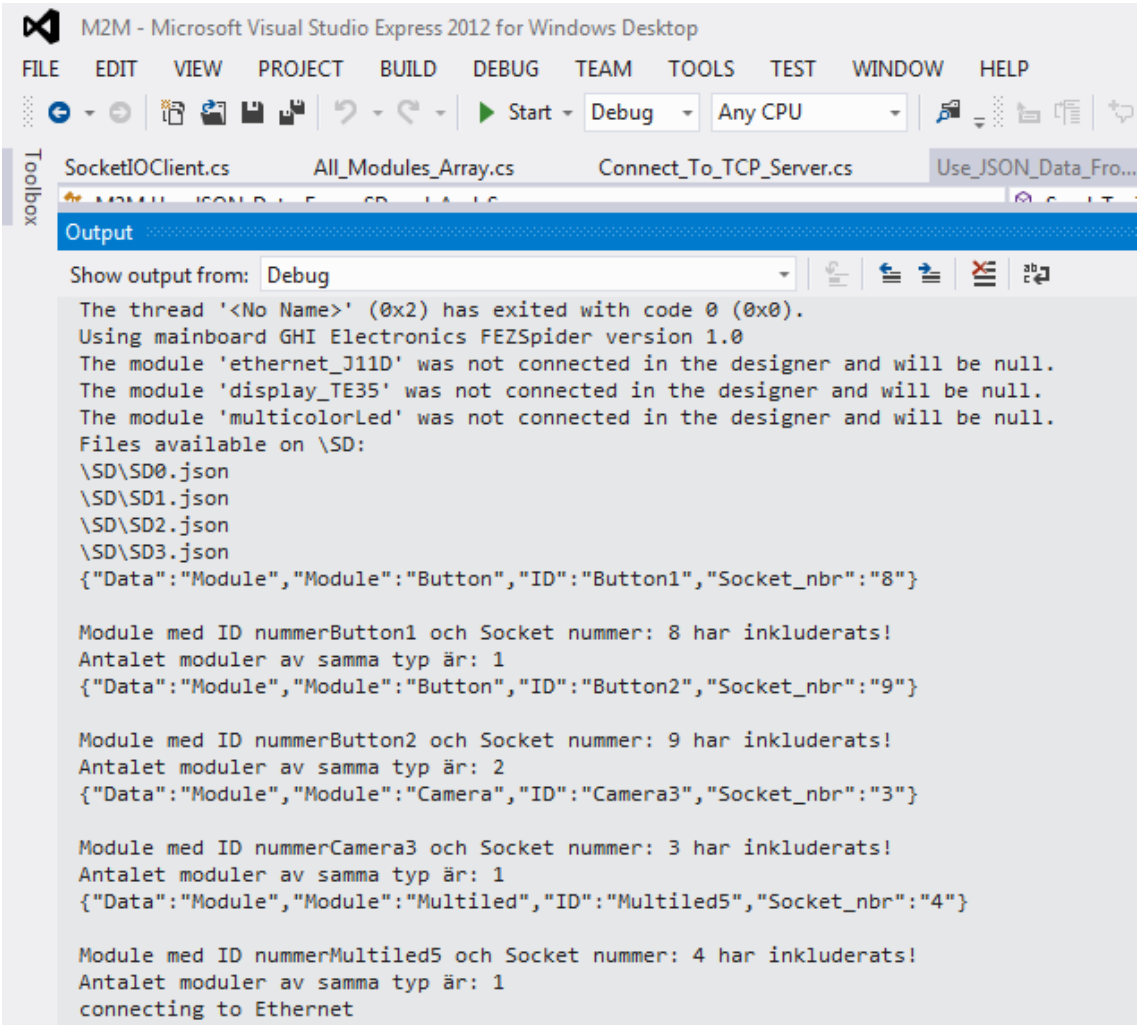


Fig 20: output M2M Visual Studio del 1

Under arbetet med TCP kommunikationen mellan Gadgeteer och servern uppkom ett problem med Socket.Connect(). texten ”Work Item 1396 - Socket.Connect() still blocked after reinsert Ethernet cable" under bug fixes i Socket.Connect(). [38] Detta ledde

till misstanke om bug på källkoden i .net micro framework SDK. Det var svårt att hitta trovärdiga källor för att bekräfta detta och den mesta informationen som hittades kommer från olika bloggar och forum.

Liknande problem hittades i flera bloggar.[39] Vid testning av koden så fastnar programmet på Socket. Connect(). Detta ledde först till slutsatsen att TCP/IP inte kan användas.

Dock efter mer testande och sökningar valdes det att använda Websockets kommunikation istället för TCP. Websockets använder sig först av http, men växlar sedan till TCP vilket betyder att det inte heller kommer att fungera om inte TCP fungerar.

Ett öppen källkod projekt användes. Projektet är baserad på websockets och som skulle kunna fungera för .netmf v4.2 som används på Gadgeteer. Samma problem uppkom dock vid SocketIOConnect (). Det var oklart om problemet berodde på samma anledning som med Socket. Connect() i TCP eftersom Websockets använder sig av TCP.

Slutsatsen blev att problemet ligger i brandväggen på datorn och inte i källkoden eller Ethernet modulen. Brandväggen hindrade kommunikationen mellan Gadgeteer och servern. Detta löstes genom att stänga av brandväggen på datorn, dock är det endast en temporär lösning då datorn utsetts för fara. Bättre lösning är att öppna en port i datorn.[40] Titta på bilaga (8.1 GHI forum). När brandväggen stängdes av fungerande kommunikation både med TCP och med Websockets.

På bilden nedan visar det att Gadgeteer har fått två nya JSON strängar där det står Got event:
'"{Data: Service, Service: LED, x: x, button: Button1}"' och
'"{Data: Service, Service: CAMERA, camera: Camera3, button: Button2}"'

```
initialized ws client!!
handshake sent
The thread '<No Name>' (0x9) has exited with code 0 (0x0).
got response:
5TmxMozt-dptz7j17w0h:60:60:websocket,htmlfile,xhr-polling,jsonp-polling
Session ID: 5TmxMozt-dptz7j17w0h
connect uri: ws://192.168.1.102:80/socket.io/1/websocket/5TmxMozt-dptz7j17w0h
ws NOT connected!
ws connected!
Connect
SocketIO connected
emitting: 5:1::{"name":"login","args":["my_identity_goes_here"]}
The thread '<No Name>' (0xa) has exited with code 0 (0x0).
Event
got event: Data'"{Data:Service,Service:LED,x:x,button:Button1}"'
The MultiLED will goes on if you press the button on socket: 8
Event
got event: Data'"{Data:Service,Service:CAMERA,camera:Camera3,button:Button2}"'
You can take a picture if you press Button on socket: 9
The Data has been  sent!
{"Data":"Module","Module":"Button","ID":"Button1","Socket_nbr":"8"}

The Data has been  sent!
{"Data":"Module","Module":"Button","ID":"Button2","Socket_nbr":"9"}

The Data has been  sent!
{"Data":"Module","Module":"Camera","ID":"Camera3","Socket_nbr":"3"}

The Data has been  sent!
{"Data":"Module","Module":"Multiled","ID":"Multiled5","Socket_nbr":"4"}

The Data has been  sent!
{"Data":"Service","Service":"LED","x":"x","button":"Button1"}
The Data has been  sent!
{"Data":"Service","Service":"CAMERA","camera":"Camera3","button":"Button2"}
ACK
```

Fig 21: output M2M Visual Studio del 2

6 Framtida utveckling

6.1 Säkerhet

Lösenord och användarnamn kan vara bra att ha i klientsidan för att hindra obehöriga att komma in i systemet. netmf stödjer också SSL (Secure Sockets Layer) säkerhetsmekanismen för att hindra tredje parten att avlyssna eller manipulera data utbyten mellan servern och Gadgeteer. [41]

6.2 Förbättring av Websockets kommunikation mellan Gadgeteer och servern

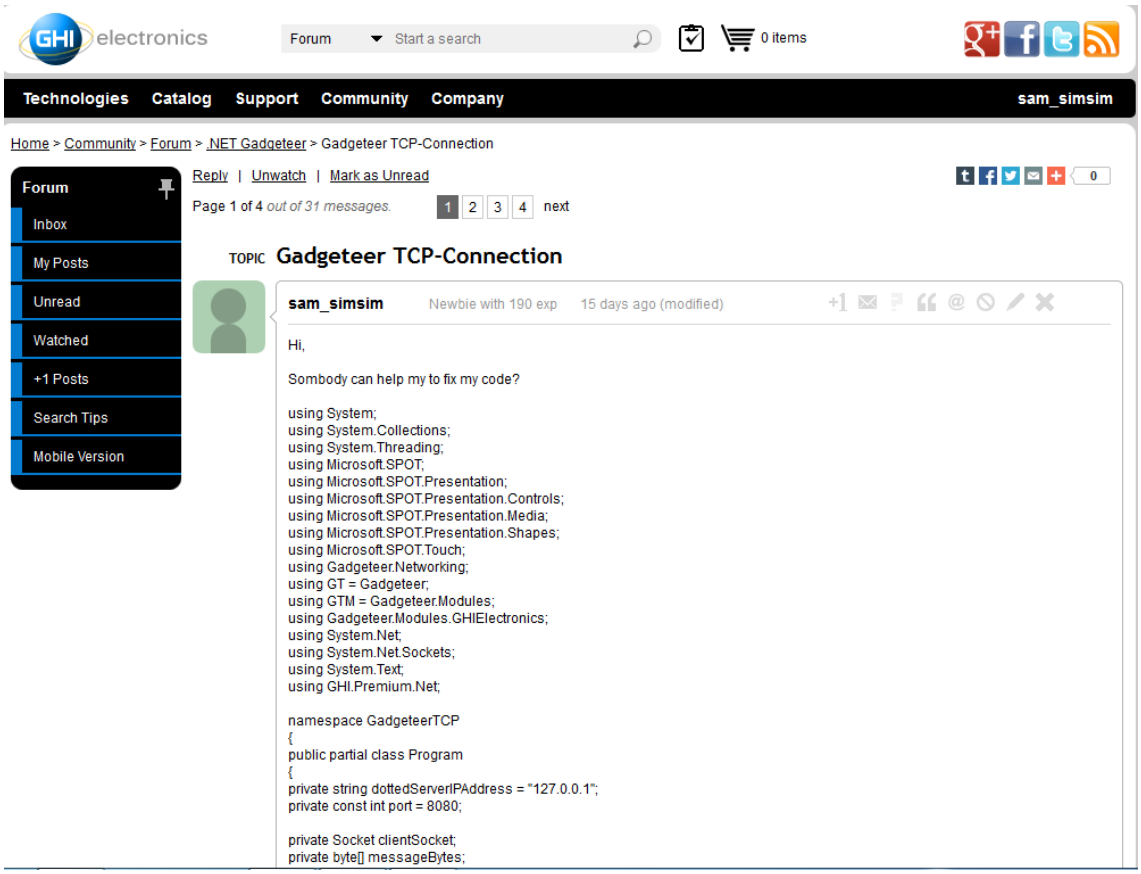
Kommunikationen mellan Gadgeteer och servern bryts ibland. Det gör att hela systemmet blir ostabilt. Detta kan bero på att Socket.io projektet är ostabilt från början eller att det inte är tillräkligt anpassat till systemmet. Lösningen på problemet är att skriva eget protokoll eller att hitta felet.

6.3 Fler funktioner på websidan

Websidan kommunicerar med Gadgeteer för att lägga till alternativ ta bort en modul eller service. Dock finns det ingen funktion som kommunicerar direkt med modulerna genom att till exempel skriva ett meddelande på websidan som visas direkt på LCD skärmen på Gadgeteer.

7 Bilagor

7.1 GHI forum



```
// This method is run when the mainboard is powered up or reset.
void ProgramStarted()
{
ethernet_J11D.DebugPrintEnabled = true;
ethernet_J11D.UseDHCP();
ethernet_J11D.UseThisNetworkInterface();

ethernet_J11D.NetworkUp += new GTM.Module.NetworkModule.NetworkEventHandler(ethernet_NetworkUp);


}

void ethernet_NetworkUp(GTM.Module.NetworkModule sender, GTM.Module.NetworkModule.NetworkState state)
{
initethernetConnection();

}



void initethernetConnection()
{
Debug.Print("connecting to Ethernet ");
if (ethernet_J11D.Interface.IsOpen)
{
Debug.Print("interface was open");
}
else
{
Debug.Print("interface was not open");
ethernet_J11D.Interface.Open();
}

ethernet_J11D.Interface.CableConnectivityChanged += new EthernetBuiltIn.CableConnectivityChangedEventHandler((s, e) =>
{
Debug.Print("Ethernet conn changed!");
if (e.IsConnected) { Debug.Print("ETHERNET connected!"); }
else { Debug.Print("Ethernet disconnected.."); }
});



var settings = ethernet_J11D.NetworkSettings;

Debug.Print("-----------------------------------------------------");
//Debug.Print("MAC: " + ByteExten.ToHexString(settings.PhysicalAddress, "-"));
Debug.Print("IP Address: " + settings.IPAddress);
```

```
Debug.Print("DHCP Enabled: " + settings.IsDhcpEnabled);
Debug.Print("Subnet Mask: " + settings.SubnetMask);
Debug.Print("Gateway: " + settings.GatewayAddress);
Debug.Print("-----------------------------------------------");


dottedServerIPAddress = settings.IPAddress;
using (clientSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream,
ProtocolType.Tcp))
{
IPEndPoint RemoteHost = new IPEndPoint(Dns.GetHostEntry(dottedServerIPAddress).AddressList[0], port);
clientSocket.Connect(RemoteHost);

messageBytes = Encoding.UTF8.GetBytes("Hello World!");
clientSocket.Send(messageBytes);
byte[] inBuffer = new byte[100];
int count = clientSocket.Receive(inBuffer);
char[] chars = Encoding.UTF8.GetChars(inBuffer);
string str = new string(chars, 0, count);
Debug.Print(str);

}




}
}
}
```

I get this error :

```
#### Exception System.Net.Sockets.SocketException - CLR_E_FAIL (1) ####
#### Message:
#### Microsoft.SPOT.Net.SocketNative::connect [IP: 0000] ####
#### System.Net.Sockets.Socket::Connect [IP: 001d] ####
#### GadgeteerTCP.Program::initethernetConnection [IP: 00e2] ####
#### GadgeteerTCP.Program::ethernet_NetworkUp [IP: 0005] ####
#### Gadgeteer.Modules.Module+NetworkModule::OnNetworkEvent [IP: 004d] ####
#### System.Reflection.MethodBase::Invoke [IP: 0000] ####
#### Gadgeteer.Program::DoOperation [IP: 001a] ####
#### Microsoft.SPOT.Dispatcher::PushFrameImpl [IP: 0054] ####
```

John GHI Employee 15 days ago #1

@sam_simsim - The 10054 error code usually indicates that the remote host closed the connection. Are you sure that the server is functioning properly?

sam_simsim Newbie with 190 exp 15 days ago #2

Hi John,

I have an node.js server. I have tesed the server and it works with a node.js and netmf client but only on gadgeteer i get this error :(.

the code for the server:

```
var net = require('net');


var PORT = 8080;



net.createServer(function(sock) {


console.log('CONNECTED: ' + ':'+ sock.remotePort);


sock.on('data', function(data) {

console.log('DATA ' + ': ' + data);

// Write the data back to the socket, the client will receive it as data from the server
sock.write('You said "' + data + '"');


});



}).listen(PORT);
```

John GHI Employee 14 days ago #3

@sam_simsim - Can you try to connect to other servers and see if the issue remains? Can you use wireshark to see if what the device is sending and receiving?

Architect Superhuman with 150,161 exp 14 days ago #4

@sam_simsim - Welcome to the forum.

Just want to point out that we have code formatting in the forum posts. This will make you post so much better. It is a button with '101010' icon. You can always go back to your published message and edit it.

sam_simsim Newbie with 190 exp 14 days ago #5

I wrote three servers but all in node.js do you mean another platform ? which one ? I never used wireshark can you please guide me ?

sam_simsim Newbie with 190 exp 14 days ago #6

I tried to connect my gadgeteer to a netmf server on C#. I get the same error. I tried to connect the server with netmf client and it worked perfectly.

Patrick Senior with 3,194 exp 13 days ago 1 #7

Hi,

Looking at your code, aren't you trying to connect to your own embedded device? I think thats the reason for the socket error you're receiving.

sam_simsim Newbie with 190 exp 13 days ago #8

Hi Patrick,

where you see that? I run same code on netmf and it's working fine.

```
private const string dottedServerIPAddress = "127.0.0.1";
private const int port =8080;

public static void Main()
{


using (Socket clientSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream,
ProtocolType.Tcp))
{
// Addressing
IPAddress ipAddress = IPAddress.Parse(dottedServerIPAddress);
IPEndPoint serverEndPoint = new IPEndPoint(ipAddress, port);

// Connecting
Debug.Print("Connecting to server " + serverEndPoint + ".");

clientSocket.Connect(serverEndPoint);
Debug.Print("Connected to server.");

// Sending

byte[] messageBytes = Encoding.UTF8.GetBytes("Hello World!");
clientSocket.Send(messageBytes);

byte[] inBuffer = new byte[100];
int count = clientSocket.Receive(inBuffer);
char[] chars = Encoding.UTF8.GetChars(inBuffer);
string str = new string(chars, 0, count);


}// the socket will be closed here
}
}


} [code=cs][/code]
```

Brett Superhuman with 84,564 exp 13 days ago #9

Sam, can you edit your posts and fix up the code tags ?

He sees this line in your code:
Code Language: C#

```
dottedServerIPAddress = settings.IPAddress;
```

that means it'll attempt to connect to itself.

ut of 31 messages. 1 2 3 4 next

vatch | Mark as Unread

0

Patrick Senior with 3,194 exp 13 days ago #10

> **Brett**
>
> Sam, can you edit your posts and fix up the code tags ?
>
> He sees this line in your code:
> **Code Language: C#**
>
> ```
> dottedServerIPAddress = settings.IPAddress;
> ```
>
> that means it'll attempt to connect to itself.

Change the IPAddress (and Port) in that line to the server running the socket listener and it should be fine.

sam_simsim Newbie with 190 exp 13 days ago #11

Hi Brett,

I chaged the code like this:

```
public partial class Program
{
private string dottedServerIPAddress = "127.0.0.1";
private const int port = 8080;

private Socket clientSocket;
private byte[] messageBytes;
// This method is run when the mainboard is powered up or reset.
void ProgramStarted()
{
ethernet_J11D.DebugPrintEnabled = true;
ethernet_J11D.UseDHCP();
ethernet_J11D.UseThisNetworkInterface();

ethernet_J11D.NetworkUp += new GTM.Module.NetworkModule.NetworkEventHandler(ethernet_NetworkUp);


}

void ethernet_NetworkUp(GTM.Module.NetworkModule sender, GTM.Module.NetworkModule.NetworkState state)
{

using (clientSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream,
```

```
{
IPEndPoint RemoteHost = new IPEndPoint(Dns.GetHostEntry(dottedServerIPAddress).AddressList[0], port);
clientSocket.Connect(RemoteHost);

messageBytes = Encoding.UTF8.GetBytes("Hello World!");
clientSocket.Send(messageBytes);
byte[] inBuffer = new byte[100];
int count = clientSocket.Receive(inBuffer);
char[] chars = Encoding.UTF8.GetChars(inBuffer);
string str = new string(chars, 0, count);
Debug.Print(str);

}

}

}
}
```

but i still get same error and if i change the IP Address to 192.168.xx it's just hangs.

Patrick Senior with 3,194 exp 13 days ago #12

Within Visual Studio place a breakpoint at the line below and look at the value of Remote Host.

sam_simsim

Code Language: C#

```
IPEndPoint RemoteHost = new IPEndPoint(Dns.GetHostEntry(dottedServerIPAddress).AddressLi
```

sam_simsim Newbie with 190 exp 13 days ago #13

The value is {127.0.0.1:8080}.

the server have same IP and Port. :/

Patrick Senior with 3,194 exp 13 days ago #14

Sam, just for the record: you are aware of the fact that 127.0.0.1 is a local loopback address and it is used to test communication on same local device?

You must replace the 127.0.0.1 ip address with the ip address of your device which is listening for incoming sockets. Start debugging from there

sam_simsim Newbie with 190 exp 12 days ago #15

I'm just confused because first you say if i use dottedServerIPAddress = settings.IPAddress; so it's mean that i want to connect to my own device then you say i need to have the IP-adress for the device ? the method settings.IPAdress generate the IP-addess for the device and it's generate 192.168.1.5 for this time.

sam_simsim Newbie with 190 exp 12 days ago #16

I have now the value like this {192.168.1.5:8080}, and the node.js server listening on port 8080. but i have the same error:

Exception System.Net.Sockets.SocketException - CLR_E_FAIL (5)
Message:
Microsoft.SPOT.Net.SocketNative::connect [IP: 0000]
System.Net.Sockets.Socket::Connect [IP: 001d]
GadgeteerTCP.Program::Interface_NetworkAddressChanged [IP: 0076]
GHI.Premium.Net.NetworkInterfaceExtension::NetworkChangeExtension_NetworkAddressChanged [IP: 0021]
GHI.Premium.Net.NetworkChangeExtension::OnNetworkChangeCallback [IP: 00ac]
GHI.Premium.Net.NetworkChangeExtension+NetworkChangeExtensionListener::OnEvent [IP: 000d]
Microsoft.SPOT.EventSink::EventDispatchCallback [IP: 0014]
SocketException ErrorCode = 10054
SocketException ErrorCode = 10054
A first chance exception of type 'System.Net.Sockets.SocketException' occurred in Microsoft.SPOT.Net.dll
SocketException ErrorCode = 10054
An unhandled exception of type 'System.Net.Sockets.SocketException' occurred in Microsoft.SPOT.Net.dll

Patrick Senior with 3,194 exp 12 days ago (modified) #17

Sam, sorry for the confusion  it is not my intention!

Let me try to get things clear

Your remote device is running the socket server (receive application) to which you want to connect from your netmf device, right? Suppose your remote device is having ip address 192.168.1.100 and your netmf is having 192.168.1.5 as ip address.

sam_simsim Newbie with 190 exp 12 days ago #18

Thank you Patrick :). I will try that.

sam_simsim Newbie with 190 exp 12 days ago (modified) #19

Patrick I have two services now but both works on localhost :/. I don't know how to make them to work for my gadgeteer. They just listning on port and any IP- address who want to connect with the server. I tried to listen on 192.168.1.5 wich is the gadgeteer IP-address, but the node.js server does not want to work and gave me an error. but it works fine if it's listning on 127.0.0.1(localhost). the other one is a netmf server and it listen on any socket too listeningSocket.Bind(new IPEndPoint(IPAddress.Any, port)), and when I try to write listeningSocket.Bind(new IPEndPoint(Dns.GetHostEntry(dottedServerIPAddress).AddressList[0], port)) it gives my an error :/.

node.js server:

```
var net = require('net');

var HOST = '127.0.0.1';
var PORT = 8080;

var client = new net.Socket();
client.connect(PORT, HOST, function() {

console.log('CONNECTED TO: ' + HOST + ':' + PORT);
// Write a message to the socket as soon as the client is connected, the server will receive it as message from the client
client.write('I am Chuck Norris!');

});

// Add a 'data' event handler for the client socket
// data is what the server sent to this socket
client.on('data', function(data) {

console.log('DATA: ' + data);
// Close the client socket completely
client.destroy();

});

// Add a 'close' event handler for the client socket
```

```
client.on('close', function() {
console.log('Connection closed');
});
```

netmf server:

Code Language: C#

```
public class Program
        {
            private const int port = 8080;
            public static void Main()
            {

            using (Socket listeningSocket = new Socket(AddressFamily.InterNetwork,
                        SocketType.Stream,
                        ProtocolType.Tcp))
             {
                     listeningSocket.Bind(new IPEndPoint(IPAddress.Any, port));
                     Debug.Print("Listening for a client...");
                    listeningSocket.Listen(1);
            using (Socket communicationSocket = listeningSocket.Accept())
            {
                Debug.Print("Connected to client.");
                //wait infinitely to get a response
                communicationSocket.ReceiveTimeout = -1;
                byte[] inBuffer = new byte[1000];
                int count = communicationSocket.Receive(inBuffer);
                string message = new string(Encoding.UTF8.GetChars(inBuffer));
                Debug.Print("Received '" + message + "'.");
              }
             }
        }
    }
}
```

Brett Superhuman with 84,564 exp 12 days ago #20

127.0.0.1 is not a valid IP address. You can not use that. It is a loopback address, a way to fudge connecting to a network on the same device, without ever really transiting out onto the network.

You have what I assume is a PC, running the node.js service, correct ? And you have implemented a connection from your PC to the node.js service, using 127.0.0.1, correct, and that all works as expected?

Your netmf device is 192.168.1.5.

You now need the 192.168.1.x address that your PC responds to. That is the device address you need as your target to go from the netmf device. If you don't do that then you will never connect.

But there's a good chance that your port 8080 is being blocked by the firewall, it is not something that is always enabled by default.

code tags. [_code] goes at the start of your code, the closing [_/code] goes at the end. You can use the pencil icon to edit the posts and fix it up.

sam_simsim Newbie with 190 exp 12 days ago #21

Hi Brett,I'm new at how server and client work.

if port 8080 is not so good wich one may i use?
PC- IP address= 192.168.1.100
Gagdeteer IP address= 129.168.1.5

the server exists on my PC so I need to connect the gadgeteer to the PC ? So I mean sockett. connect going to be like this:

socket.connect("192.168.1.100", 8008)
and the server will listen on IP address = 0.0.0.0 and port=8080 wlch means it will listen on all IP addresses who want to connect to port 8080.

Is it correct ?

Brett Superhuman with 84,564 exp 11 days ago #22

In general yes what you're thinking is correct. The "server" has to listen in for any inbound connection on a defined port. The client needs to open the connection to the destination IP address on that port.

8080 may not be a good choice because it may not be configured in your firewall as an inbound port - any port you choose may not make sense, you really need to know what you're doing here and put in appropriate rules / exclusions in your firewall software, if you're not using something like a standard web server and it's associated port, port 80. If you have a 2nd PC you could test that to prove that you can get to the server, rather than trying to debug two unknowns at the same time.

sam_simsim Newbie with 190 exp 11 days ago #23

Tank you guys! my Gadgeteer works now . However I still have a little issue with the port. I have to turn off the firewall to make it work. I'm using port 80. Any suggestions?

Patrick Senior with 3,194 exp 11 days ago #24

Great that you've got it working!!

As regarding to the firewall, if your PC and netmf device are at your home network and your home network is protected by a router with firewall you can disable the firewall on your PC. That way you can develop and start testing and eliminate all external factors. You can deal with those factors when you're done developing.

Keep up the development

Brett Superhuman with 84,564 exp 11 days ago #25

I wouldn't take quite that lenient a view - the firewall is there to protect you against things you don't know are happening to any network, so prolonged running without it is unhealthy. What OS are you running ?

Patrick Senior with 3,194 exp 11 days ago (modified) #26

@brett: Just for testing purposes, as soon as things are working as they should, you should bring it one step further and start configuring your firewall.

btw I'm using a Mac with OSX and vmware fusion

Brett Superhuman with 84,564 exp 11 days ago #27

yes, sorry, I should have clarified. My question about the OS was for @sam_simsim.

sam_simsim Newbie with 190 exp 11 days ago #28

I have windows 7

Brett Superhuman with 84,564 exp 11 days ago #29

http://windows.microsoft.com/en-us/windows/open-port-windows-firewall?woldogcb=0#1TC=windows-7

8 Referenser

8.1 Referenser

[1] Hodges, S.; Taylor, S.; Villar, N.; Scott, J.; Bial, D.; Fischer, P.T.. Prototyping connected devices for the internet of things. Computer, 2013, 46(2):26-34. DOI: 10.1109/MC.2012.394

[2] Data Ductus. 2014. about Data Ductus. http://dataductus.se/about/. (Hämtad: 2014-11-18).

[3] Data Ductus. 2014. Machine to Machine/M2M. http://dataductus.se/services/m2m/. (Hämtad: 2014-11-18).

[4] Senta, S.; Johnston, S.; Hodges, S.; Scott, J.; Schwiderski-Grosche S.; Kuçera J.(.u.å.) Learning to Program with Visual Basic and .Net Gadgeteer.

[5] Hodges, Steve. Time for Gadgeteer. Apr2013, Vol. 50 Issue 4, p20-21. 2p. DOI: 10.1109/MSPEC.2013.6481689.

[6] Jeff _Winn .2013. .NET Gadgeteer Core 2.42.700. Apr 29. https://gadgeteer.codeplex.com/releases/view/105366 . (Hämtad: 2014-11-18).

[7] Kühner, Jens. 2009. Expert .NET Micro Framework. 2 uppl. Berkeley, CA.

[8] Alan, Thorn. 2008. Cross Platform Game Development. Chap 5.P137.Plano, Texas. (Hämtad: 2014-11-18).

[9] Microsoft. Developer Network. API References for .Net Micro Framework. (.u.å.) http://msdn.microsoft.com/en-us/library/hh401281.aspx . (Hämtad: 2014-11-18).

[10] Microsoft. Extensible Emulator. (.u.å.) http://msdn.microsoft.com/en-us/library/ee433256.aspx . (Hämtad: 2014-11-19).

[11] Microsoft. Introducing the Extensible Emulator. (.u.å.) http://msdn.microsoft.com/en-us/library/ee433277.aspx . (Hämtad: 2014-11-19).

[12] Microsoft. Developer Network. 1. Introduction(C#). (.u.å.) http://msdn.microsoft.com/en-us/library/aa645597%28v=vs.71%29.aspx . (Hämtad: 2014-11-18).

[13] Mats, Lilja. Programmering i C++.EDA623, Något om C#(Föreläsning14). 2013. http://fileadmin.cs.lth.se/cs/Education/EDA623/Forelasningar/cpp_fe.pdf . (Hämtad: 2014-11-18).

[14] Dave Gandy. 2014. NuGet Gallery Home. http://www.nuget.org/ . (Hämtad: 2014-11-18).

[15] Xavier, Decoster. 2013. An Overview of the NuGet Ecosystem. http://www.codeproject.com/Reference/628210/An-Overview-of-the-NuGet-Ecosystem . (Hämtad: 2014-11-18).

[16] NuGet. NuGet Overview.(.u.å.). http://docs.nuget.org/docs/start-here/overview . (Hämtad: 2014-11-18).

[17] Cory, Gackenheimer. 2013. Node.js Recipes. A Problem-Solution Approach. 1uppl.

[18] George Ornbo.2012. Teach Yourself Node.js in 24 Hours. 1uppl. USA.

[19] Microsoft. Telnet: Vanliga frågor och svar. (.u.å.). http://windows.microsoft.com/sv-se/windows/telnet-faq#1TC=windows-7 . (Hämtad: 2014-11-18).

[20] JSON. Introducing JSON. (.u.å.). http://www.json.org/ . (Hämtad: 2014-11-18).

[21] Matt, Weimer. Json.NetMF. (.u.å.). https://github.com/mweimer/Json.NetMF . (Hämtad: 2014-11-18).

[22] Nurseitov, N.; Paulson, M.; Reynolds, R.; Izurieta, C..Comparison of JSON and XML Data Inter change Formats: A Case Study. (.u.å.) . http://www.cs.montana.edu/izurieta/pubs/caine2009.pdf . Bozeman, Montana. (Hämtad: 2014-11-18).

[23] Margaret Rouse. Search Networking. TCP (Transmission Control Protocol). (.u.å.) http://searchnetworking.techtarget.com/definition/TCP . (Hämtad: 2014-11-18).

[24] Fette,I.; Melnikov, A..2011. The WebSocket Protocol. http://tools.ietf.org/html/rfc6455 .

[25] Kaazing. About HTML5 WebSockets. (.u.å.). https://www.websocket.org/aboutwebsocket.html. (Hämtad: 2014-11-18).

[26]Cody Lindley, Package Managers: An Introductory Guide For The Uninitiated Front-End Developer, https://tech.pro, http://tech.pro/tutorial/1190/package-managers-an-introductory-guide-for-the-uninitiated-front-end-developer#front_end_developers . (Hämtad: 2014-12-16).

[27] MDN, JavaScript, https://developer.mozilla.org/en-US/docs/Web/JavaScript. (Hämtad: 2014-12-16).

[28] MDN, Introduction, A re-introduction to JavaScript (JS tutorial), file:///C:/Users/Default.Dell-E6410/Desktop/sidor/A%20re-introduction%20to%20JavaScript%20%28JS%20tutorial%29%20-%20JavaScript%20_%20MDN.htm . (Hämtad: 2014-12-23).

[29] Margaret Rouse, techtarget, http://searchsoa.techtarget.com/definition/HTML . (Hämtad: 2014-12-16).

[30] Spiros Mancoridis, intruction to the hypertext marc-up language (HTML), file:///C:/Users/Default.Dell-E6410/Desktop/sidor/html.pdf . (Hämtad: 2014-12-23).

[31] MCD, CSS, https://developer.mozilla.org/en-US/docs/Web/CSS . (Hämtad: 2014-12-16).

[32] Webdesignskolan,information om CSS, http://webdesignskolan.se/css/css.php#info . (Hämtad: 2015-01-01).

[33] 3scale.2011. 3scale Networks S.L. What is an API? Your guide to the Internet Business (R) evolution. http://www.3scale.net/wp-content/uploads/2012/06/What-is-an-API-1.0.pdf . (Hämtad: 2014-11-19).

[34] Jacobson D.; Brail G.; Woods D.; 2011.APIs: A Strategy Guide 1uppl. Sebastopol, CA.

[35] Alirezaei, R.; Schwartz,B.; Ranlett, M.; Hillier, S.; Wilson, B.;Fried, J.; Swider, P.. 2013. Professional SharePoint 2013 Development. Chap 17.P 653.1 uppl.

[36] jasdev55. 2013. WebSocket Client for .NET and .NETMF. http://jdiwebsocketclient.codeplex.com/ . (Hämtad: 2014-11-18).

[37] Niko Mäkitalo. 2013. SocketIO.NetMF. https://github.com/nikkis/SocketIO.NetMF. (Hämtad: 2014-11-19).

[38] CodePlex. 2012. .NET MF 4.3 (RTM). https://netmf.codeplex.com/releases/view/81000 . (Hämtad: 2014-11-19).

[39] Gavin Osborn. 2011. Socket Connect - a word of warning. http://www.dotnetsolutions.co.uk/blog/socket-connect---a-word-of-warning . (Hämtad: 2014-11-19).

[40] Windows Help. Open a port in Windows Firewall . http://windows.microsoft.com/en-us/windows/open-port-windows-firewall?woldogcb=0#1TC=windows-7 .(Hämtad: 2014-11-20).

[41] Microsoft Developer Network, Implementing SSL Connections, http://msdn.microsoft.com/en-us/library/jj646587%28v=vs.102%29.aspx . (Hämtad: 2014-11-28).