|  |
| --- |
| **Graduation Project** |
| Mobile Based Remote Control |
|  |
| Introducing GSM system as a communication channel in a Real-Time Remote Control and Monitoring System |
|  |
| **Salah Aldeen Ghanem**  **Supervisor: Dr.**  **Samer Mialle** |
| **5/6/2011** |
|  |

Contents

[Project Overview 2](#_Toc292915383)

[Introduction 3](#_Toc292915384)

[Purpose 3](#_Toc292915385)

[Scope 4](#_Toc292915386)

[System Architecture 4](#_Toc292915387)

[Project Software 5](#_Toc292915388)

[2.1 Mobile Software 5](#_Toc292915389)

[Symbian OS 5](#_Toc292915390)

[Android 7](#_Toc292915391)

[2.2 Server Software 13](#_Toc292915392)

[2.3 PC Software 13](#_Toc292915393)

[2.4 Microcontroller Code: 14](#_Toc292915394)

[Project Hardware 15](#_Toc292915395)

[A-Arduino Nano board 15](#_Toc292915396)

[B-7805 Voltage Regulator 16](#_Toc292915397)

[C- DC motors 16](#_Toc292915398)

[D- Servo motor 16](#_Toc292915399)

[E-Temperature Sensor 17](#_Toc292915400)

[F- Bluetooth mate 17](#_Toc292915401)

[G- H-Bridge 18](#_Toc292915402)

[Project Parts Prices: 18](#_Toc292915403)

[Appendix (Adopted Protocols) 19](#_Toc292915404)

[TCP/IP 19](#_Toc292915405)

[Bluetooth Protocols 20](#_Toc292915406)

[Radio frequency communication (RFCOMM) 20](#_Toc292915407)

[Logical link control and adaptation protocol (L2CAP) 20](#_Toc292915408)

# Project Overview

## Introduction

This project proposes a model of using GSM System as a part of a Real-Time Remote Control and Monitoring System.

Advantages of such a model lies behind the facts of the availability of mobile networks and devices, since there are more than 1000 Mobile network around the globe with over than 3.5 billion subscribers and network coverage of 99% of earth’s urban areas. Not mentioning the internet speed, reliability and the cost effectiveness of using mobile packet access feature provided by both GSM and UMTS Networks.

Other big advantage is the reliability of mobile networks and their high uptime (usually above 98% uptime), and not forgetting the reliability of internet in general which makes this system valid even for critical applications,

One more point is the great potentials of Mobile handhelds and their availability made it more than possible to use them in such application.

## Purpose

We are introducing a modern control system that uses the internet provided by Mobile Networks (both GSM and UMTS) to connect the machine desired to be controlled to a PC application that can run on any PC anywhere as long as it has internet connectivity.

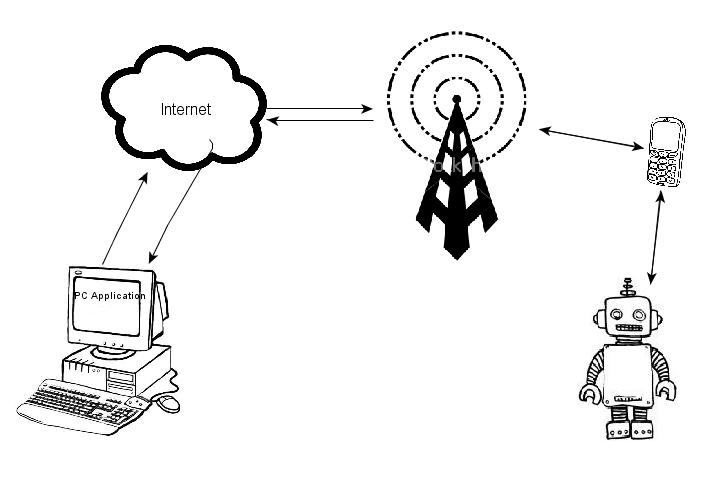


Fig 1.1

On the machine side I used Bluetooth Link between the mobile phone and the Microcontroller controlling the machine which in this case was a "Toy Car", this way the mobile phone can control several machines simultaneously from distance, which allows using this model in controlling factories and collecting data from acquisition systems and remote sensors.

## Scope

This model future depends on the way it is meant to be used, if it is to be used for Handsets that is carried by persons (not limited for control) this will limit its capabilities but provide a user friendly solution which is perfect for smart houses and SAN applications.

In this approach this model can be adopted in the Palestinian market since Mobile Service Providers (Jawwal &Wataniya**wataniyawataniyawataniyawataniyawataniyawataniyawataniya**) already uses an (EDGE) data packet access protocol with a bandwidth of at least 8Kbps. This is more than enough for sending and receiving control signals.

On the other hand it can be developed to be specified for the control system which allows editing the mobile OS and avoid all its limitations; which became possible due to the fact that Google made their Mobile OS (Android) open source and licensed it under GNU GPL license which allows editing, viewing and redistributing of the code.

Now using a phone as a part of a control system will allow you to do live video and audio streaming from the mobile to the PC, this requires a significant increase in the bandwidth needed, and this can be done over a 3G network using one of its High Speed Packet Access (HSPA) protocols.

This option also can target the Palestinian market, since Wataniya Company has a license for operating a 3G network in Palestinian.

## System Architecture

This project is divided into four main parts:

1. PC Application,
2. Server Application,
3. Mobile Application and sensors,
4. Microcontroller Code and the rest of Hardware.

The microcontroller board (arduino) is connected to a control circuit (H-Bridges) of the car DC motors and throw analog ports to sensors (Temperature Sensor), and connected at the same time “serially” throw (UART port) to a Bluetooth module.

On the mobile side, the mobile application takes readings, frames from mobile sensors and camera and communicates with both microcontroller (throw Bluetooth) and PC application throw TCP/IP sockets, which is a full-duplex communication channel.

At the user end (PC application) the application handle user inputs and displays the data and images received from mobile.

One last thing is the server application which runs on an internet server that keeps track of mobile IP and PCs IP which provides the promised portability for the PC application.

# Project Software

## 2.1 Mobile Software

The lack of experience and the weakness of mobile development sector in Palestine drove me to spend months not knowing what exactly shall I do, nor how to get what I want. What I needed is a performance mobile application that can run all the time, alone (without human interaction) and interface the camera and other hardware and able to control them.

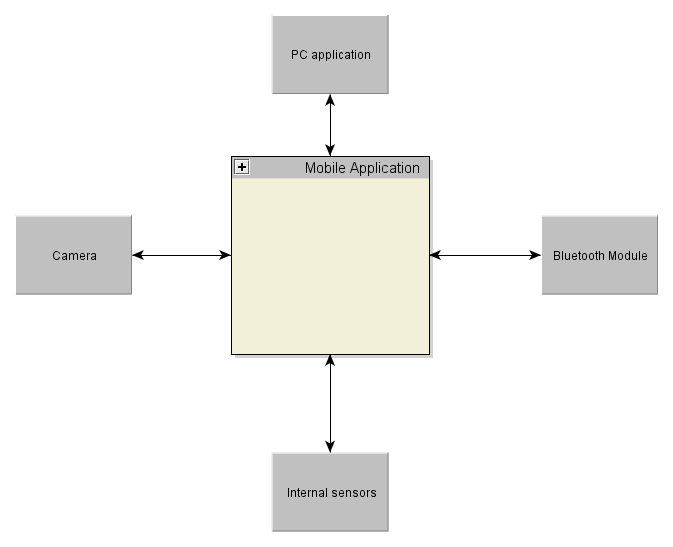


Fig 2.1

### Symbian OS

At the beginning I used Nokia’s Symbian S60 platform; since symbian Mobile Operating System shares more than 40% of the global market as statistics of the 4Q of 2008 I thought it would be best to use it.

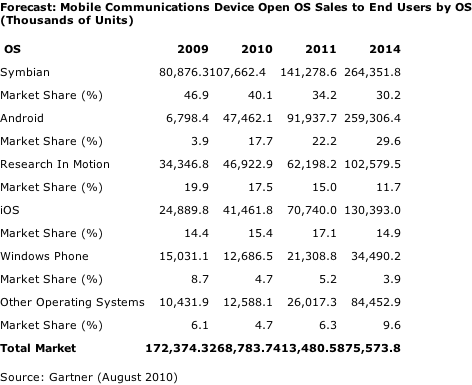


Table 2.1

The previous table shows the market share of mobile OSs and how symbian is dominating, although it’s losing its reputation for other smart phones.

The problem was that Nokia’s symbian OS runs on poor performance mobile devices that can’t handle heavy background applications like ours for ex. Below the hardware specifications of Nokia N95 which I thought can run my app (since it is one of the best Nokia devices).

##### Nokia N95 Specifications:

1- Dual ARM9 CPU (322 MHz),

2-32 M.B ram,

3- Symbian OS V9.2.

But after doing allot of tests on this mobile I realized that the poor ram and the low internal storage made it hard to have such application actually working on such environment.

That was on hardware level; on software level there were three options to program for Symbian S60 OS:

1. Native Symbian C/C++ ,
2. Java ME,
3. Python for S60 (PyS60).

The first option was inefficient because it doesn’t provide direct accessibility to mobile hardware (away from the OS) and can’t overcome security limits such as the need to ask for user permission every time the app tries to access the hardware (camera, Bluetooth …) still it was very hard to code.

I didn’t give much time for Java ME because it wasn’t clear enough (no enough documentations) and buggy, still its major defect was the fact it was running on a virtual machine (JVM) that doesn’t have enough resources because of the system design and the low available ram.

At last the python solution which is a famous scripting language ported to S60 OS to provide developers a fast way to prototype their application. So it’s not made to run performance critical applications. And that was proved by experience. I.e. my application always crashes in less than 5 minutes and it was a small part of what I wanted to do.

To sum up Symbian OS can’t be used for heavy duty applications because of:

1. Poor Hardware (RAM and CPU power),
2. Lake of APIs and resources.
3. Bad Security System approach.

So I needed to search for alternatives.

### Android

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Google Inc. purchased the initial developer of the software, Android Inc., in 2005. Android's mobile operating system is based on the Linux kernel. Google and other members of the Open Handset Alliance collaborated on Android's development and release. The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android. The Android operating system is the world's best-selling Smartphone platform by 2011.

In 2007 80 Hardware, Software and Telecom companies including (Google, HTC, Sony, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung, LG, T-Mobile, Nvidia, and Wind River Systems) formed Open Handset Alliance (OHA) which is the industrial skeleton of Android OS.

##### Samsung Galaxy S Specifications:

1. 1 GHz ARM Cortex A8 (1.5GHz if Overclocked).
2. 512 MB RAM
3. 128 MB GPU
4. Android 2.2
5. 2.6.32.9 Linux Kernel

Applications for android are written in Java and can contain native code (C++) in this case developer needs to write his own JNI libraries or written completely in native C++. And since android is based on Linux kernel allot of User space Linux applications can be ported to android environment.

#### Android Security System

Android is a privilege-separated operating system, in which each application runs with a distinct system identity (Linux user ID and group ID). Parts of the system are also separated into distinct identities. Linux thereby isolates applications from each other and from the system.

Additional finer-grained security features are provided through a "permission" mechanism that enforces restrictions on the specific operations that a particular process can perform, and per-URI permissions for granting ad-hoc access to specific pieces of data.

#### Application Components

Application components are the essential building blocks of an Android application. Each component is a different point through which the system can enter an application. Not all components are actual entry points for the user and some depend on each other, but each one exists as its own entity and plays a specific role—each one is a unique building block that helps define the application's overall behavior.

There are four different types of application components. Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed.

Here are the four types of application components:

Activities: An activity represents a single screen with a user interface.

Services: A service is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. Another component, such as an activity, can start the service and let it run or bind to it in order to interact with it.

Content providers: A content provider manages a shared set of application data. You can store the data in the file system, an SQLite database, on the web, or any other persistent storage location the application can access. Through the content provider, other applications can query or even modify the data (if the content provider allows it).

Broadcast receivers: A broadcast receiver is a component that responds to system-wide broadcast announcements. Many broadcasts originate from the system—for example, a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured. Applications can also initiate broadcasts.

In my application I used Activities and Services and Broadcast receivers. The following chart describes the life cycle of both:

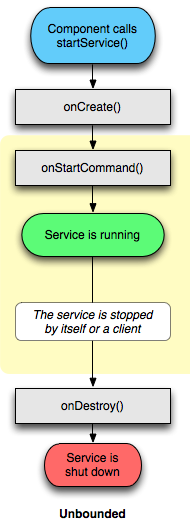
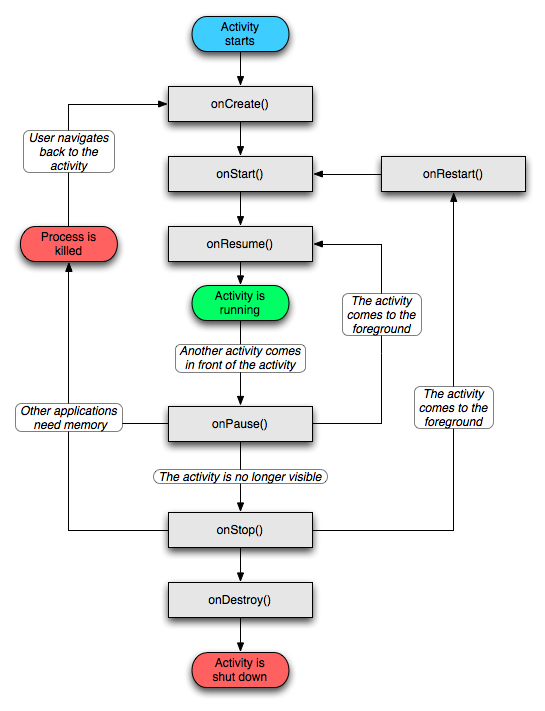


Fig 2.2

#### Development Environment

There are two main development tools for android environment

1. Android Native Development Kit (NDK) for writing native binaries and JNI libraries.
2. Eclipse IDE with android SDK for writing Java applications.

#### Code Description

My code consists of the following classes:

* Main.java: an activity used for starting the application, even thought it will start automatically after the mobile boots up, but it can be used if anything went wrong to restart the service.. Also it has a TCP server related to it and spawned in a deferent thread so it doesn’t block preview images rendering on that surface.
* CamView.java: an activity used to create a valid surface to render camera preview frames on, and it has a native method to compress preview images from their color space (YUV) to JPG image format and pushes them to a stack.
* StackManager.java: a class contains synchronized methods to push and pop images from a linked list that behaves as a stack (Last In First Out), the need for synchronized methods because we needed a thread safe way to push images from CamView thread and pop them from CamServer thread.
* CamServer.java: a class implements an application specific TCP server which sends the image data to the PC application according to the following protocol.

Fig 2.3

* Serv.java: a Service that represents our application body, it listens for PC application connection and starts other classes and manages them.
* Status.java: a class that listens for Battery, GPS, GSM and internal sensors and provides interface to get feedback to the application about them and to control its listeners.
* Bluetooth.java: a class that manages the Bluetooth connection between the car and the mobile throw creating a Bluetooth SPP client socket that connects to the SPP server socket on the module side.
* TCPServer.jav a class implements an application specific TCP server which sends data and feedback messages to the PC application according to the following protocol.

Fig 2.4

#### Application Life Cycle

The following flow chart describes the Mobile application behavior.

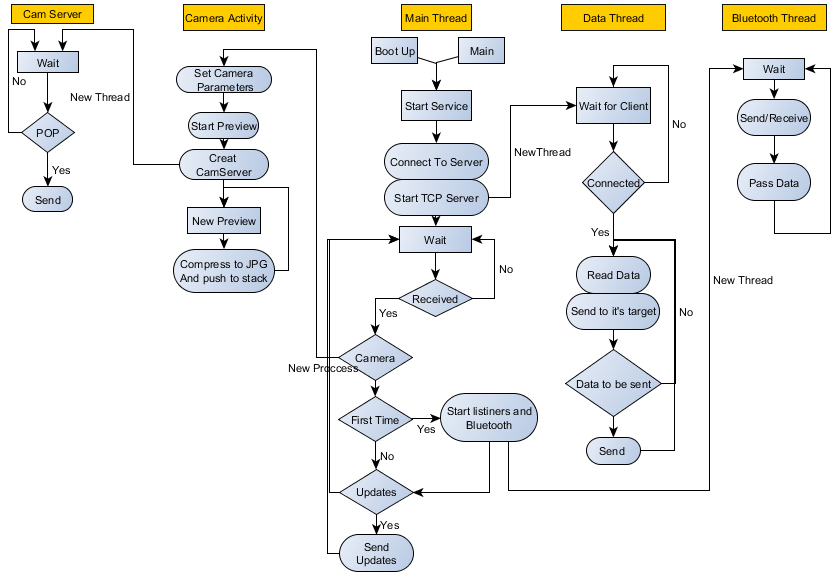


Fig 2.5

\*note that this isn’t the exact behavior of the application, due to Java’s object oriented nature.

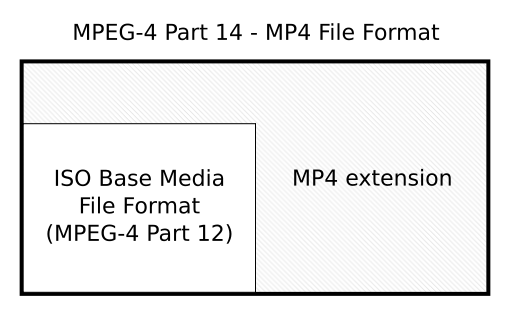
#### Live Video Streaming

Since day one I was trying to get a real video stream out of my mobile phone (can be played with any media player) first on symbian, it wasn’t possible due to the fact that symbian OS is a limiting environment (You only do what you’re allowed to as a developer), and the sever lack of CPU and ram power to do live streaming.

One of the reasons made me choosing android is its ability to do live streaming (there are too many apps in the market doing live video streaming such as SIPdroid, IMSDroid, Ustream, …) the first two apps are open source VoIP clients that implements video calls.

##### MediaRecorder API

I spent allot of my time researching video streaming in android, first I tried to output a Video Recorded by android system API (MediaRecorder) to a TCP socket directly since in Linux a TCP socket can be handled same as files (except it’s unseekable), which didn’t work out because of the following



In multimedia there is a part of a Multimedia file contains data about video encoding, length, voice encoding and extra information, this part is called Metadata, and it’s supposed to be in the file header, now when MediaRecorder Class starts recording it doesn’t know how long will the record continues so it can’t write the metadata when record starts, instead it writes incorrect bytes all over the metadata part, and once done recording, it seeks back the file beginning and writes the correct metadata;

Fig 2.6

Since we want live streaming, metadata shall be available at the beginning of the stream which can’t be done with MediaRecorder class.

Still Sipdroid writes the video data to local socket and then adds metadata manually to the video header and restreams it.

##### MJPEG

MJPEG is a standard streamable video format, which is basically a sequence of JPG images, used in webcams and CCT cameras, the simplest implementation is a web server on the mobile and listens for clients, and when the client connects it send jpg images separated by a standardized tag.

This method can be done, but it is less efficient from what we are doing, since we are streaming JPGs over plain socket (not using HTTP protocol).

##### FFMPEG

FFmpeg is a very rich and famous open source Linux based multimedia application, that can be used as a shared library. FFmpeg was ported to arm architecture at the beginning of this year. It can be used to encode, decode media files in many formats, and has a sub application “FFserver” which can be used as multimedia streaming server.

On a Linux PC FFmpeg can interface V4l2 library (Video 4 Linux 2) which manages the webcam, get YUV images from it, encode it and send it to ffserver application.

I tried to do so on my Galaxy because it has V4L2 library as the following chart shows:

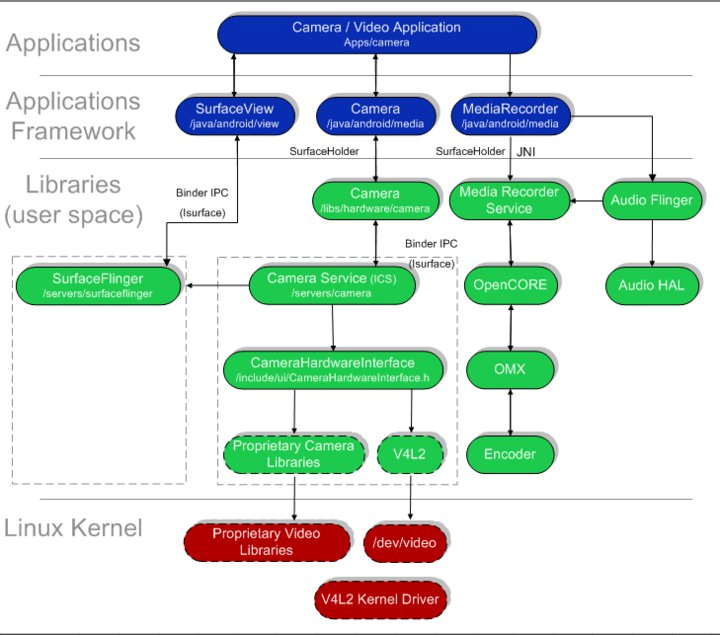


Fig 2.7

But unfortunately every time I try to access /dev/video camera the device reboots instantaneously, I don’t know exactly what caused this, but most likely due to the unclean porting of ffmpeg which causes severe load on the system and force it to reboot.

Another way was to pass uncompressed (YUV) or compressed frames (JPGs) to ffmpeg and then encode them as H.264 and pass them to ffserver, this method also can work, and it is a better solution but:

1. My FFmpeg bin was not a clean build and when I used it to read JPGs and encode them the average CPU load was about 2 which is totally unacceptable.
2. I couldn’t find a way to pass images directly to ffmpeg without writing them to the SDcard, while the writing process takes about 200ms for each frame, this method went inefficient.

#### Performance issues

Even though android mobiles are considered high performance devices, still the OS is optimized to run as many applications as possible with the lowest cost of CPU power and RAM usage, that’s why android has its own Garbage Collector implementation, which blocks the application and swap out all inaccessible code and variables, sometimes it takes GC about 200ms to get its job done and might interferes many times in one second, so it’s very crucial for applications that are performance based to have an as much clean as possible code, thus android SDK has a code optimization tool, which helped allot in reshaping the code to have a cleaner solution the result was: GC interferes my code every 5 seconds and takes about 30ms .

## 2.2 Server Software

Many European and American companies sells VPSs (Virtual Private Server) for cheap prices, some of them as cheap as 2$/month, these servers usually runs a Linux PC or Server distribution and runs 24 7. So using one of them for keeping track of cars IP changing is a perfect option.

I already have one of them and so I wrote a simple java application that does the following

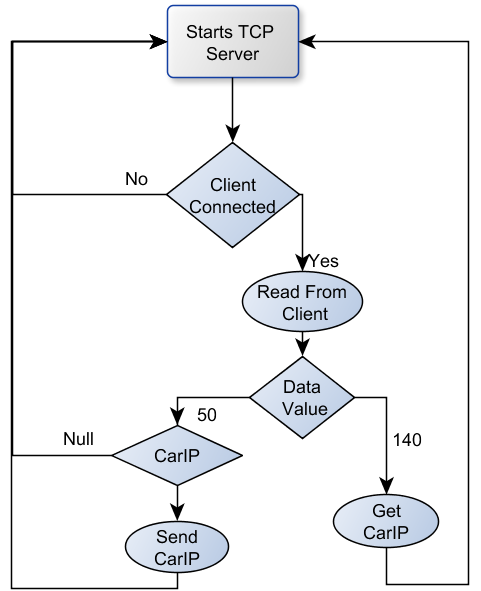


Fig 2.8

## 2.3 PC Software

The PC Software is written in C# Programming Language which is an object-oriented programming language developed by Microsoft as part of their .NET initiative, and later approved as a standard by ECMA and ISO. C# has a procedural, object-oriented syntax based on C++ that includes aspects of several other programming languages (most notably Delphi and Java) with a particular emphasis on simplification.

The PC application consists of:

1. A Form Class which provides the user interface ( buttons, keyboard inputs, displays),
2. A Data Client Class which is a class has a TCP client responsible for sending and receiving data and commands from the PC to the car and vice versa,
3. An Image Client Class which is a class has a TCP client responsible for sending camera commands and receives Images from the Mobile.

The PC application also is a multithreaded application where image, data clients runs in their own threads away from the main thread which is the Form in our case.

## 2.4 Microcontroller Code:

I used Arduino Nano board for this project, which is an open source development board, based on Atmel AVR microcontroller, it can be programmed in a C like language and it is optimized for prototyping since it saves allot of time writing codes that already wrapped into its supported libraries.

Analog

Digital I/O

ADC

UART

Fig 2.9

It has a ready to use servo control library and another library for initializing and handling UART port

# Project Hardware

The project hardware consists of:

## A-Arduino Nano board

Arduino is an open-source single-board microcontroller, descendant of the open-source Wiring platform, designed to make the process of using electronics in multidisciplinary projects more accessible. The hardware consists of a simple open hardware design for the Arduino board with an Atmel AVR processor and on-board I/O support. The software consists of a standard programming language compiler and the boot loader that runs on the board.

Arduino hardware is programmed using a Wiring-based language (syntax + libraries), similar to C++ with some simplifications and modifications, and a Processing-based IDE.

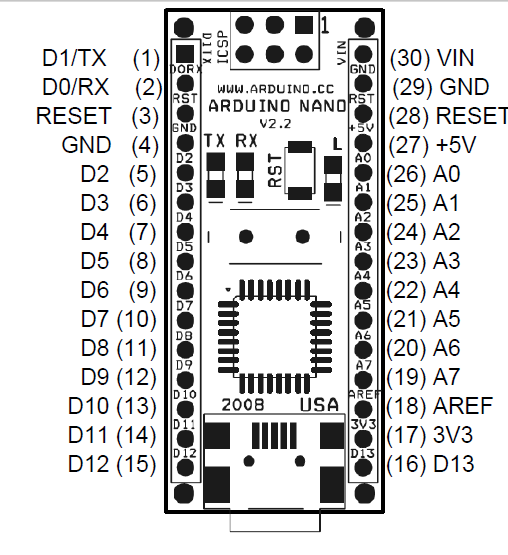


Fig 3.1

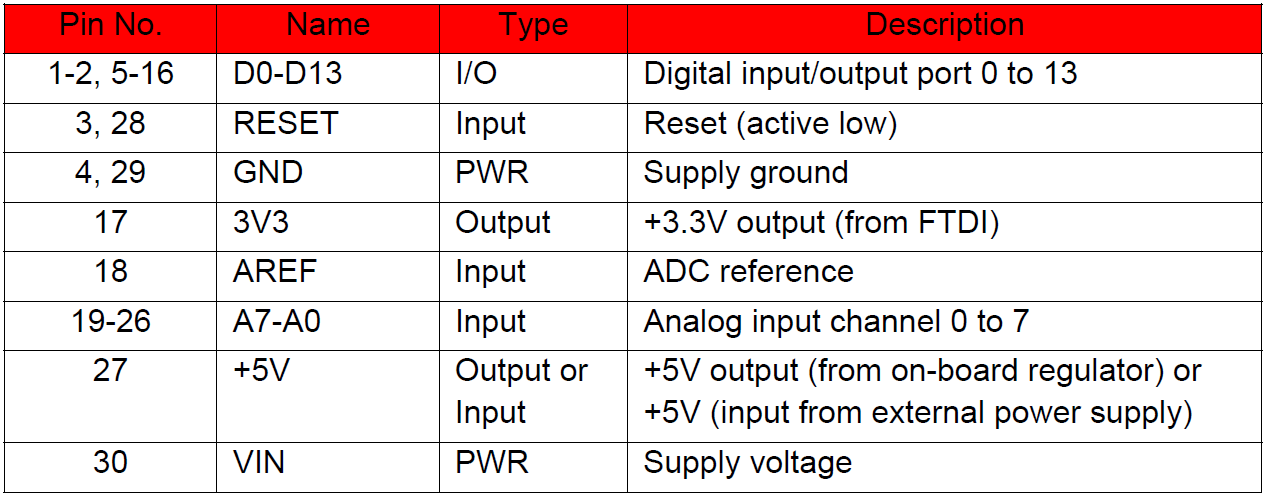
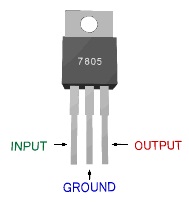


Table 3.1

## B-7805 Voltage Regulator



*The 78XX series is a positive voltage regulation series, each IC has its own heat sin, current limiter, and short circuit protection, it provides a maximum current of 1.5Amps which is more than enough for running our IC.*

Figure 3.2

|  |  |
| --- | --- |
| IC | Current drain@ 5V |
| Arduino Nano 2.2 | 200 mA |
| LM35dz | Less than 60µA |
| L293 | 150 mA |
| Total | Less than 400 mA |

Table 3.2

## C- DC motors

DC motors represents a suitable solution for wheel driving, because:

1. Of their small size/torque ratio which makes them weight efficient,
2. Their (Direction and Speed) Control circuits are simple,
3. And they are cost efficient option.



|  |  |
| --- | --- |
| Characteristics | Value |
| VDC | 4.2-7.5 |
| Amps | 150 mA |
| Speed | 2400 RPM |

Table 3.2 Figure 3.2

## D- Servo motor

Servo motors are special DC motors with position feedback used in applications where angle matters most, in our case we are using them to rotate the mobile phone (which is mounted on the car) to get a better viewing angle, or receiving better GPS signal.

Features:

* Continuous 360° rotation
* Rest point adjustment

|  |  |
| --- | --- |
| Characteristics | Value |
| Operating voltage | 4.8-6.0VDC |
| Torque | 3.3-4.8 kg/cm |
| Speed | 60-70RPM |

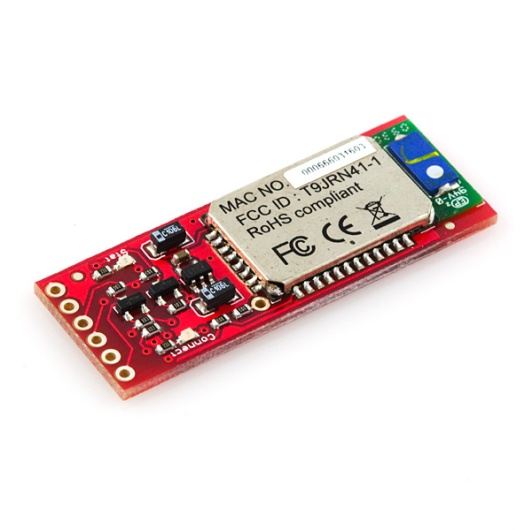
Table 3.3 Figure 3.3

## E-Temperature Sensor

The LM35 series are precision integrated-circuit temperature sensors, which has an output voltage that is linearly proportional to the Celsius (Centigrade) temperature,

V (out) = 10.0 mV/°C

|  |  |
| --- | --- |
| Characteristics | Value |
| Operating voltage | 4-30 V |
| current drain | 60 μA |
| Accuracy | 0.5°C guaranteed (at +25°C) |

Table 3.4 Figure 3.4

## F- Bluetooth mate

*Bluetooth mate is a simple Bluetooth modem consists of Roving Networks RN-41 Bluetooth module, with a built in voltage regulator, and protections against short circuit.*

Figure 3.5

Features:

* supports many Bluetooth versions (2.1,2.0,1.2,1.1),
* Low power consumption (250uA deep sleep, <10mA sniff mode, 30mA connected),
* UART (SPP or HCI) interface,
* High bit Rate (300kbps master ,240kbps slave),
* Embedded Bluetooth stack profile with (GAP, SDP, RFCOMM, and L2CAP protocols, with SPP and DUN profile support).

## G- H-Bridge

L293 dual H-Bridge IC, used to drive the DC Motors,

Features

* 4-36 V input range,
* 600mA max current for each channel,
* Thermal shutdown,
* TTL-compatible inputs.

# Project Parts Prices:

|  |  |  |
| --- | --- | --- |
| Part | Amount | Price(NIS) |
| Samsung Galaxy S I9000 | 1 | 1700 |
| Bluetooth mate | 1 | 260 |
| Arduino Nano | 1 | 160 |
| 6V DC-Motor | 2 | 16 |
| LM35dz | 1 | 4 |
| 7805 | 1 | 5 |
| L293 | 1 | 10 |
| Servo Motor | 1 | 81 |
| Battery | 1 | 60 |

Table 3.3

# Appendix (Adopted Protocols)

## TCP/IP

The Transmission Control Protocol (TCP) is one of the core protocols of the Internet Protocol Suite. TCP is one of the two original components of the suite, complementing the Internet Protocol (IP), and therefore the entire suite is commonly referred to as TCP/IP. TCP provides the service of exchanging data directly between two hosts on the same network, whereas IP handles addressing and routing message across one or more networks. In particular, TCP provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer. TCP is the protocol that major Internet applications rely on, applications such as the World Wide Web, e-mail, and file transfer. Other applications, which do not require reliable data stream service, may use the User Datagram Protocol (UDP) which provides a datagram service that emphasizes reduced latency over reliability.

TCP provides a point-to-point channel for applications that require reliable communications.

TCP provides a communication service at an intermediate level between an application program and the Internet Protocol (IP). That is, when an application program desires to send a large chunk of data across the Internet using IP, instead of breaking the data into IP-sized pieces and issuing a series of IP requests, the software can issue a single request to TCP and let TCP handle the IP details.

IP works by exchanging pieces of information called packets. A packet is a sequence of octets and consists of a *header* followed by a *body*. The header describes the packet's destination and, optionally, the routers to use for forwarding until it arrives at its destination. The body contains the data IP is transmitting.

Due to network congestion, traffic load balancing, or other unpredictable network behavior, IP packets can be lost, duplicated, or delivered out of order. TCP detects these problems, requests retransmission of lost data, rearranges out-of-order data, and even helps minimize network congestion to reduce the occurrence of the other problems. Once the TCP receiver has reassembled the sequence of octets originally transmitted, it passes them to the application program. Thus, TCP abstracts the application's communication from the underlying networking details.

TCP is optimized for accurate delivery rather than timely delivery, and therefore, TCP sometimes incurs relatively long delays (in the order of seconds) while waiting for out-of-order messages or retransmissions of lost messages. It is not particularly suitable for real-time applications such as Voice over IP. For such applications, protocols like the Real-time Transport Protocol (RTP) running over the User Datagram Protocol (UDP) are usually recommended instead.

## Bluetooth Protocols

### Radio frequency communication (RFCOMM)

The Bluetooth protocol RFCOMM is a simple set of transport protocols, made on top of the L2CAP protocol, providing emulated RS-232 serial ports (up to sixty simultaneous connections to a Bluetooth device at a time). The protocol is based on the ETSI standard TS 07.10.

RFCOMM is sometimes called serial port emulation. The Bluetooth serial port profile is based on this protocol.

RFCOMM provides a simple reliable data stream to the user, similar to TCP. It is used directly by many telephony related profiles as a carrier for AT commands, as well as being a transport layer for OBEX over Bluetooth.

Many Bluetooth applications use RFCOMM because of its widespread support and publicly available API on most operating systems. Additionally, applications that used a serial port to communicate can be quickly ported to use RFCOMM

In the protocol stack, RFCOMM is bound to L2CAP.

### Logical link control and adaptation protocol (L2CAP)

L2CAP is used within the Bluetooth protocol stack. It passes packets to either the Host Controller Interface (HCI) or on a hostless system, directly to the Link Manager.

L2CAP's functions include:

* Multiplexing data between different higher layer protocols.
* Segmentation and reassembly of packets.
* Providing one-way transmission management of multicast data to a group of other Bluetooth devices.
* Quality of service (QoS) management for higher layer protocols.

L2CAP is used to communicate over the host ACL link. Its connection is established after the ACL link has been set up.

In basic mode, L2CAP provides packets with a payload configurable up to 64 kB, with 672 bytes as the default MTU, and 48 bytes as the minimum mandatory supported MTU. In retransmission and flow control modes, L2CAP can be configured for reliable or isochronous data per channel by performing retransmissions and CRC checks. Reliability in either of these modes is optionally and/or additionally guaranteed by the lower layer Bluetooth BDR/EDR air interface by configuring the number of retransmissions and flush timeout (time after which the radio will flush packets). In-order sequencing is guaranteed by the lower layer.

The EL2CAP specification adds an additional enhanced retransmission mode (ERTM) to the core specification, which is an improved version of retransmission and flow control modes. ERTM is required when using an AMP, such as 802.11abgn.