KVPY 2010 Technical Report

A project report on

A free messenger program that let you to send messages worldwide at no cost

Submitted by:

GURSIMRAN SINGH
Thapar university

Kvpy App - 982681

Preface

In this portion of the report I would like to add some experiences that I got while working on this project and making this report. First of all my heartiest thanks to Mr. Anil Kumar Verma who gave me time to time valuable advice since I joined the college here.

Making the report was a great experience. I have typed each word of this report myself and in my own language. Also, I have realised the importance and necessity of documentation.

Another very important thing that I have learnt while working on this project is how to do research, how to handle thing when you don't know what the result would be, how keep patience and keep working in such situations. I have realised the importance of original work and what it takes to do it. Copying something is very easy, even if you know someone has done it before, replicating it is quite easy but when you are working on something original then it is very difficult and it takes a great effort.

The choice of project is influenced by agricultural problem in Punjab that fields of farmers are far away from their homes. As I myself belong to an agriculture background family, so I got the exposure of this problem. As tube wells are limited so people have to share the time of use. People have to wake up at say 2:00am to turn ON/OFF the tube wells. This is certainly not convenient so I thought to work out this problem innovatively in this project but I will have to make the software two-way to get feedback from the tube well that it is operating correctly. I hope to install this system when I get time.

I want to emphasize this here that, not only I have a innovator aspect which is that the ability to think some easy and more beneficial way to faster some need or tackle a problem but also a researcher aspect that is to push that idea by experimentation, logic and testing to make it a reality.

To be honest, research in basic sciences and especially in physics, is my passion and deep interest. I have a very good computer background as well. I have been programming since 6^{th} grade and now I know quite well how to use this excellent tool in research to achieve great things. I want to choose research in basic sciences as a career in my further life and I hope KVPY will set the stepping stone for that.

I have included a CD along with this report that has a video in which I have demonstrated the working application.

Introduction

Communication is a very important driving force for innovation because communication is the most basic need of a human being. A mobile phone is the first thing that will come to a person's mind when he thinks of communication. Now we communicate internationally by sending calls, SMSs, MMSs GPRS and a lot of other things. All these methods cost us. Isn't there a way that is absolutely free and enables us to communicate?

Yes there is and unintentionally we all use it very frequently in our daily lives. It's a miss call and we all use it quite often in our daily lives. Many times my friend would say that if I miss call him, then do it, else forget about it.

There is a point in this miss call communication; it's absolutely free, no cost. But there is a limitation that we can transfer only one information with this. What if we can transfer some more data from simply a miss call. Isn't there a way? Yes there is!!

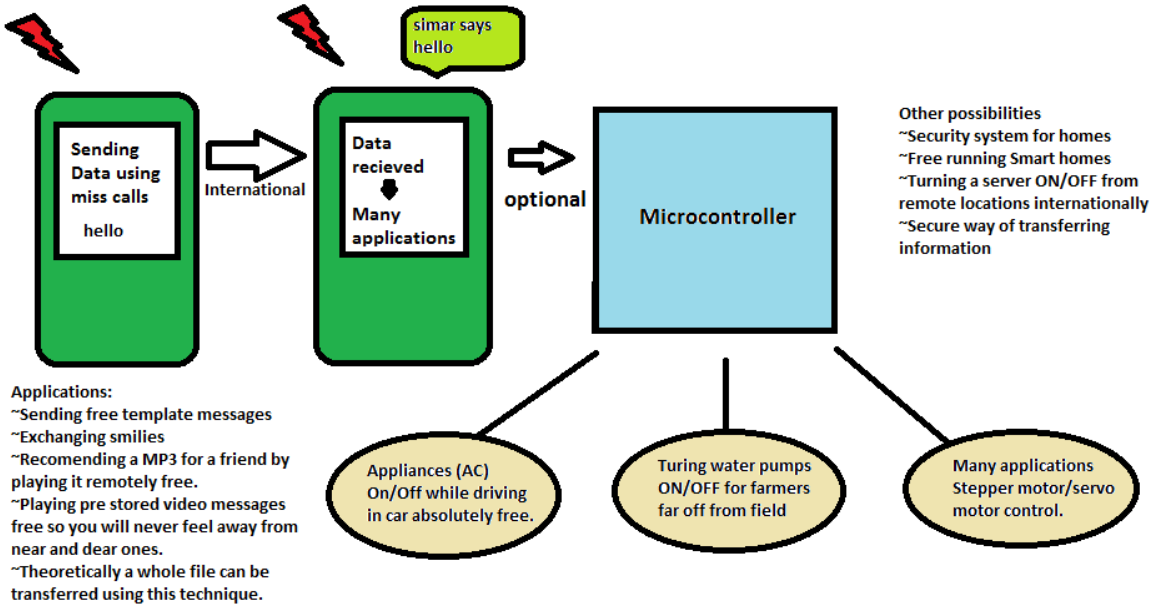
Let me explain this, if I miss call him for less than 10s then do this, if I miss call him for more than 10s then do that. Great! Now we have transferred two informations from one miss call. Still the person has to interpret the message himself so there is a big limitation and also the method is error prone. What could we do is to make this process automatic and more accurate, we need a computer program that will interpret miss call time and report us with the message associated with it. That a 'Free messenger program that lets you communicate worldwide free of cost'.

AIM of my project

My research project aims to make and test a mobile program that can interpret this miss call time and convert it to meaningful information. While working on this project for around a month I found that this 'meaningful information' can be implemented in many applications to make some great innovative products. For example you can very precisely control (ON/OFF) appliances from remote locations (worldwide), send 15 template messages worldwide totally free of cost.

Getting to the problem

Basic setup



I have organised my work based on stages. Actually I started my research journey around one month back on 9/8/10. Since then, I have written four versions of code, introducing some changes/improvements to the basic version. My aim to document the report on the basis of timeline is inspired from research like style. This way, I can share the problems faced while implementing the idea and how I tackled them.

Choice of Platform and Language

Surely I have to write code that could implement my idea. This involves sending miss call of a particular time to another phone. Java seems to be a good choice as it is widely used in phones and has a good support base on internet. However, as far as my knowledge is concerned, Java will not get required amount of access privileges to the 'calling' subsystem, so I avoided it. As my basic requirement, I should opt a platform that can give me full access to the 'calling' subsystem and also has a good API available to interact with it. It was in my knowledge already; that Symbian OS fulfils all these requirements {1}.

Symbian OS is the operating system used typically in Nokia E series and N series and some Music Express Edition smart phones. Written in C++, it contains libraries, user interfaces and frameworks that provide support to a number of development environments {2}. Excellent web based support, references and SDKs provided by Nokia makes this the ideal choice.

Due to many development platforms provided by Nokia, I had to choose which one suits my application and at the same time keep my life easy. Symbian C was the ideal choice but as my aim is just to show a basic implementation of the idea, the easiness that python offers attracted me to use. However in the end, I do realize the limitation that python has and there is no doubt that C is more

accurate in mobile development and embedded systems. I have already started working on a C solution of the above problem.

Python for S60 pyS60

Python is a very easy programming language and has a very simple syntax. I will brief about some of the modules and their functions that I used in my project. So here is a list of all the modules that I used.

- The **appuifw** module offers an interface to the S60 UI application framework
- The **e32** module offers Symbian OS related utilities that are not related to the UI and are not provided by the standard Python library modules.
- As the name suggests, the **audio** module enables recording and playing audio files and access to device text-to-speech engine.
- The **telephone** is the most important module that I have used in all programs. This provides an API to the telephone services provided by the operating system.

Although most of the code is self explanatory and I have added comments wherever possible but still I would like to add some of the important module functions that I have used in the code.

e32.ao_sleep(duration, telephone.hang_up) – This function puts the application to sleep and return the command to the OS. This way OS can run other applications for this amount time. After 'duration' the OS will call the function telephone.hang_up. This function hangs up the active call.

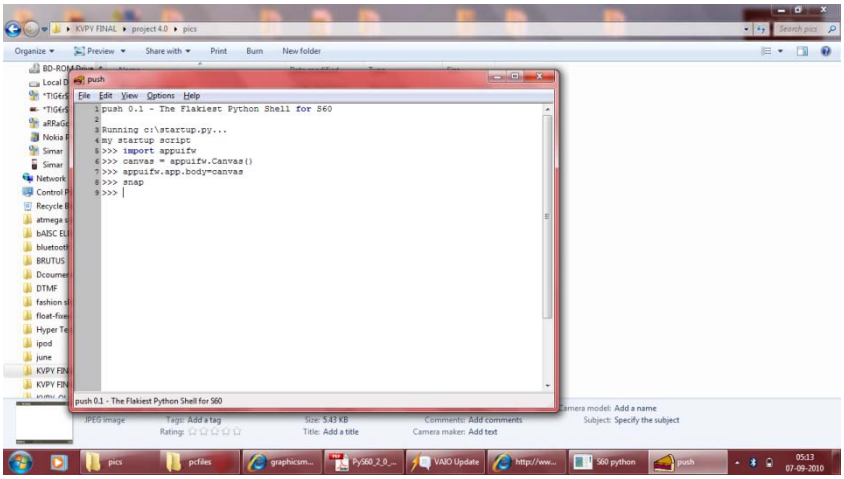
telephone.call_state(handle_hang_up) – This function of telephone module registers the function handle_hang_up and call it whenever there is a change in state of the telephone subsystem. The parameters of the function handle_hang_up(stateInformation) is a tuple, so stateinformation[0] contains the new state of telephone subsystem. {page -67 pyS60 documentation}.

telephone.incoming_call() – Make the application handle the incoming call.

canvas=appuifw.Canvas() – This creates a canvas. Canvas is one of the most popular UI in Python. As the name indicates, it provides an empty canvas on which you can draw different shapes, display images and many more.

I have included the complete python runtime environment in the CD included. It also contains the official pyS60 2.0 documentation, I have mentioned in the references above.

My Development Environment



To develop python scripts on windows pc, I installed python from official python2.7 website and pyS60 2.0 from Maemo Garage pys60 page. As my scripts include 'telephone' related services I can't test it in the emulator so I had to take help from Bluetooth console to test code snippets. After installing python a my nokia phone, I had a Bluetooth Console in computer that execute commands directly on phone using pyS60 interpreter on phone. This helped me a lot to test a lots and lots of code snippets.

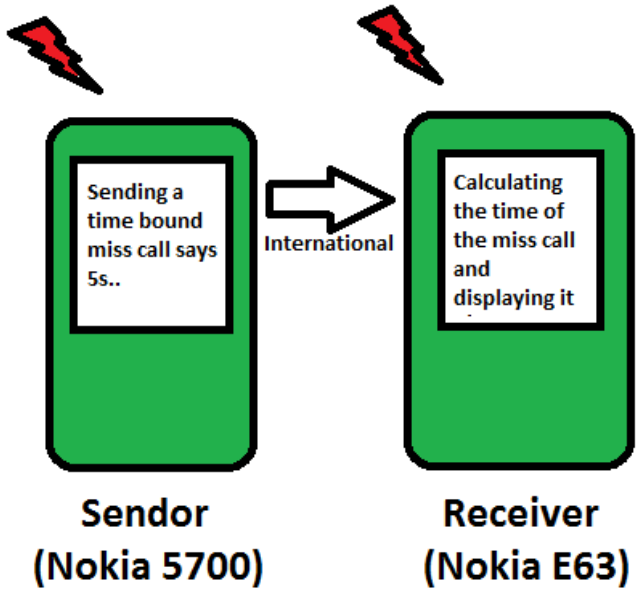
Choice of Mobile Phones

Initially I chose a old Nokia 6600 and my new E63 as my test devices but Nokia 6600 is based on S60 2nd Edition. This version of S60 is a bit old now and it does not support 'callbacks'. I tried to implement the code using free running timers in python but with no success. So I borrowed a Nokia 5700 from a friend and happily it supported callbacks so I used it in my project successfully. My choice of sender and receiver is as follows.

Sender

Receiver

Stage ONE – 20-8-2010



Initially, its implementation seemed easy but it was certainly not. As the idea of converting the miss call time into meaningful information was my own **original idea** and there is nothing on internet on this. I did not even get help for calculating miss call time from anywhere. Probably nobody cared to calculate the miss call time before this and why should they. So only after struggling a lot with callbacks, timers and clocks and using 'telephone' module, I finally succeeded in getting the time difference of the miss call at the receiver end. Here is the python script code for both receiver and sender

Receiver - This code calculates the time of miss call and displays it on the python script application.

```
import telephone
import time

print "\n\n\nSIMAR /... Heya!!!"

def newphonestate (stateInformation):

   global start_ringing
   global end_ringing

   print "\nI'm being calledback."

   newState = stateInformation[0]

   if newState == telephone.EStatusRinging:
      start_ringing = time.clock()
      print "::The new phone is ringing, call is from %s" %stateInformation[1]
      print start_ringing

   elif newState == telephone.EStatusDisconnecting:
      end_ringing = time.clock()
      print "::A call is being disconnected"
      print end_ringing

   elif newState == telephone.EStatusIdle:
            print "::Idle"
            simar()

def simar():
```

```
    print "\n\n Inside simar"
    print end_ringing
    print start_ringing

    # 'time difference' of the call
    diff = (end_ringing-start_ringing)
    print diff

#this prepares the code to handle the incoming calls
telephone.incoming_call()

#this registers the function newphonestate with the OS for callbacks
# when a change a state of 'telephone' occurs.
telephone.call_state(newphonestate)
```

Please note the '**time difference**' quoted in comments, we shall use this terminology for all further discussions.

Sender – This code sends a time restricted miss call to the number provided in the code.

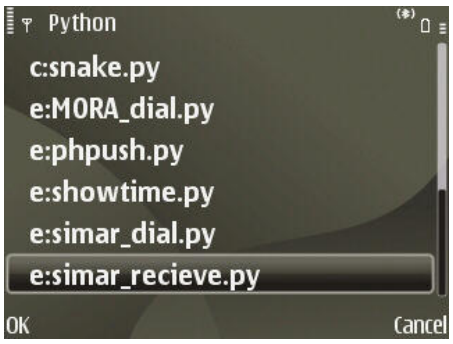
```
import telephone
import e32
import appuifw


number = 9915326619

#There will be a pop up with a input box that asks for 'call time'
duration = appuifw.query(u"Enter the Durations",'number')

telephone.dial(str(number)) #make call

#define the callback handler function
def handle_hang_up(status):
   if status[0] == telephone.EStatusConnecting:
            print "\nConnecting"
      e32.ao_sleep(float(duration), telephone.hang_up)

telephone.call_state(handle_hang_up) #set the handler function
```

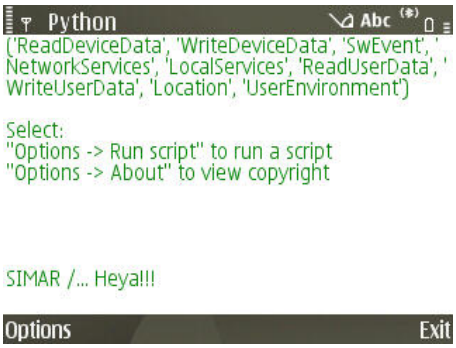
Note the quoted text '**call time**' in the comments. We shall again use this reference in the text any further.

Here is the glimpse of how things go in the mobile phones

How things go

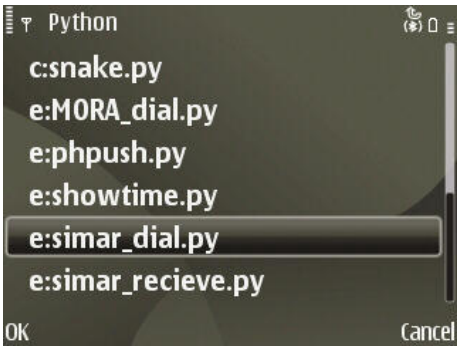
Receiver

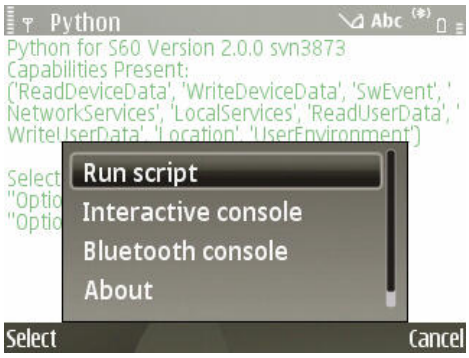


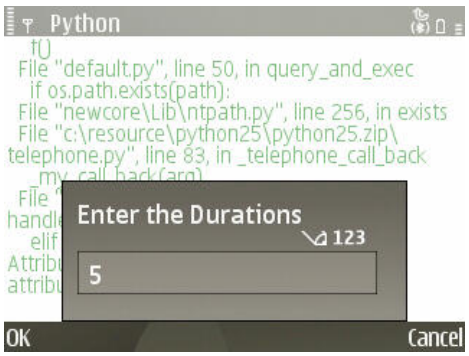


Then we use sender to send a miss call of a particular duration (5s in this case). Please note that all snapshots has been taken in the receiver mobile E63 for simplicity, However this activity actually happens in sender.

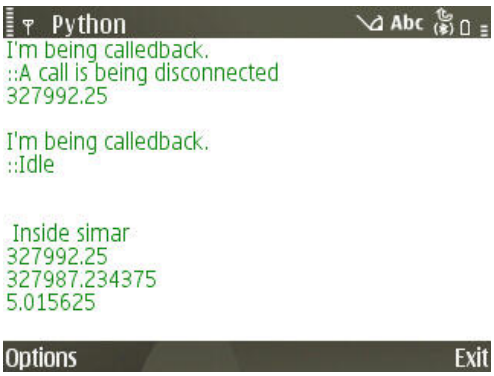
Sender







Receiver



Here 5.015 is the 'time difference'. Quite accurate in this case!!

Problem Statement

With the help of this programs I took my first set of data on 21/8/2010. The very first thought was that there could be constant offset in the values but more values, proved this wrong. After I took the values I found it was neither a constant offset not any linear relationship exists. Still there was a very light hope that the values will be precise and I can have a one-one mapping of the values. This would mean a lot of information can be sent in just one miss call. To prove this I took the values again and yes the values weren't precise as well. If I send a one sec miss call from the sender the receiver will think time of miss call as say 1.03,1.49,1.22 etc.

Data Logged

	outputs				
Data sent from senser(in sec)	21-08-2010	22-08-2010	23-08-2010	25-08-2010	26-08-2010
0	0.441	0.654	0.753	1.156	1.176
1	1.593	1.875	1.671	1687	1687
2	2625	2.671	2.687	2.812	2.64
3	3.187	3.656	3.625	3.676	3.67
4	4	4.18	5.013	4.98	4.96
5	5.17	6	6.84	6.01	5.07
6	7.04	6.983	6.933	6.893	6.554
7	8.01	7.234	8.013	7.98	7.99
8	8.95	9.01	9.03	9.4	8.98
9	9.85	10.01	9.98	10.04	9.98
10	10.96	10.93	10.1	10.98	11.23
11	11.934	12	11.7	12.01	11.88
12	13.01	12.31	12.987	13.1	12.776
13	13.06	13.93	13.98	13.65	14.011
14	15	**15.95**	**15.96**	14.88	14.67
15	16.01	15.84	**15.96**	16.2	15.402
16	16.96	16.92	16.4	17.01	17.35
17	17.96	17.57	18.04	18.2	18.33
18	19.02	18.88	18.01	19.01	19.11
19	19.76	19.89	19.66	19.923	19.725
20	20.82	20.4	20.36	21.4	21.2
21	21.8	21.96	22.02	22.39	21.8
22	22.98	22.96	23.11	23.09	22.45
23	23.67	23.89	23.54	23.19	23.86
24	24.98	24.43	24.64	25.05	25
25	25.98	25.456	25.46	25.98	25.78
26	26.69	26.98	26.4	26.59	27.1
27	27.98	27.81	27.81	28.07	28.01
28	28.56	28.46	28.91	28.45	28.79
29	29.72	29.69	29.77	29.39	29.64
30	31.03	30.89	30.37	31.14	31.21
31	31.78	31.98	31.54	31.56	31.47
32	33.45	33.23	33.23	33.01	32.98
33	33.91	33.57	33.24	33.92	34.12
34	34.78	34.41	34.73	34.86	34.33
35	35.93	35.89	36.08	36.02	35.91
36	36.66	36.85	36.97	36.36	36.84
37	37.38	37.687	37.546	37.667	37.13
38	38.956	38.567	38.68	39.2	38.646
39	39.867	39.678	39.39	39.57	39.234
40	41	40.98	41.04	41.12	40.98
41	41.91	41.91	41.87	41.63	42.15
42	42.789	42.98	42.68	42.13	42.53
43	43.65	43.09	44.01	43.657	43.132
44	44.93	44.93	44.456	44.675	44.675
45	45.13	45.968	45.89	45.21	45.35
46	user busy, range exceed..				

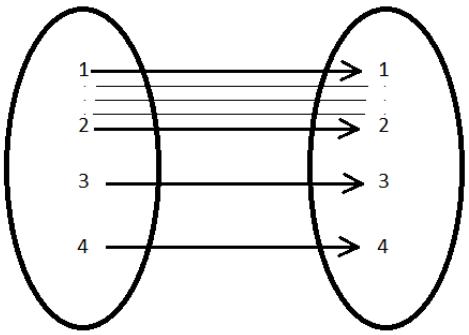
Solution

As I expected earlier the time difference was not the same when I repeatedly passed the same 'call time' value, rather fluctuating but what was a boon to me that the fluctuations were only in a particular range. This is the foundation on which my project is standing and greatest success of my project. This proved the feasibility of the idea and that it can be exploited to send information without cost. Initially I tried to implement a one-one mapping but failed.

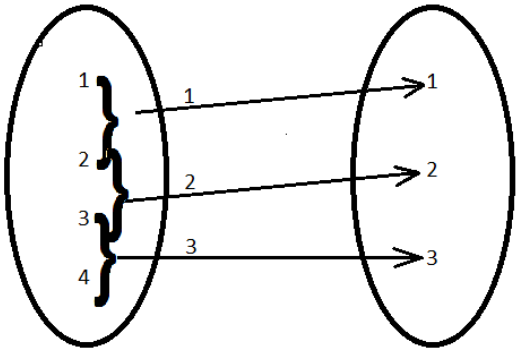
So to convert it into a workable solution I dropped the one-one mapping of data and shifted to a many-one approach. So looking into the Table the values of 0s range from 0.44 to 1.76 so I programmed all values less than 2 in receiver to correspond to 0s in sender. This way I was able to send 26 distinct values.

To test the solution I distributed the programs to some of my friends and they reported errors in information transferred. As sending more values is not my aim at this spot of time and stability in information transfer or precision is much more important I choose to be on a safer side by grouping the values in groups of *three* that correspond to only one value in receiver. This was my stage TWO.

Initial implementation

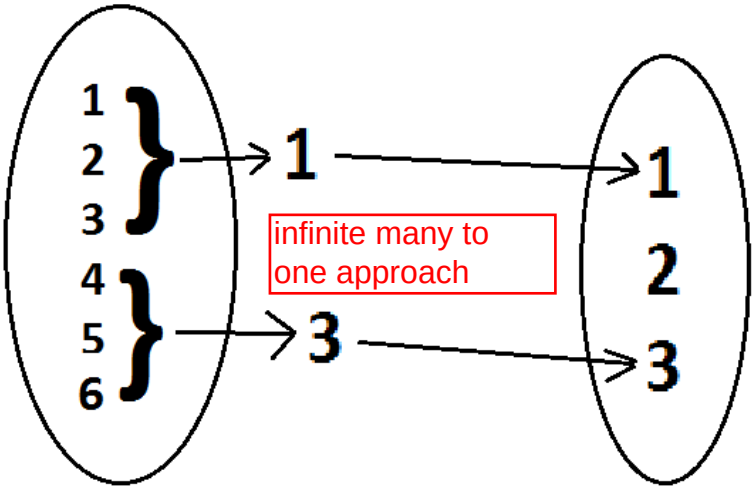


one-one mapping

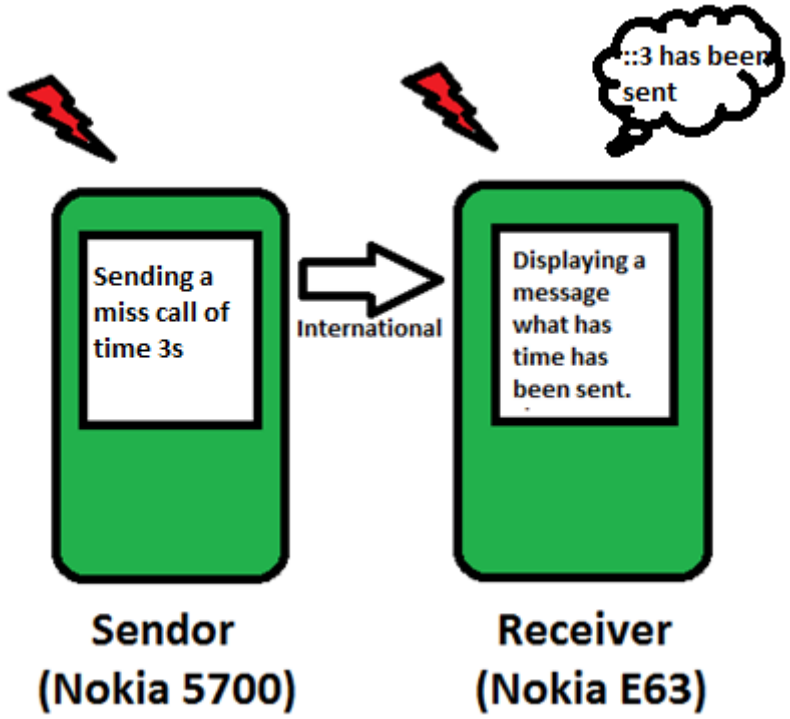


Many-to one mapping

Final implementation



Stage TWO – 26-8-2010



As, in stage one I have got time difference that is not very precise and based on the testing done in stage one I had decided to implement 3 is to 1 mapping of data. Although this will decrease the number of distinct information that I can send to 15 only but as I stated earlier my aim is to make the protocol more precise to show a basic implementation of the idea.

Receiver - This code calculates the time of miss call, interpret the 'time difference' as meaningful information and displays it on the screen.

```
import telephone
import time
import e32
import appuifw

print "\n\n\nSIMAR /... Heya!!!"

def newphonestate (stateInformation):

    global start_ringing
    global end_ringing

    print "\nI'm being calledback."

    newState = stateInformation[0]

    if newState == telephone.EStatusRinging:
        start_ringing = time.clock()
        print "::The new phone is ringing, call is from %s" %stateInformation[1]
        print start_ringing

    elif newState == telephone.EStatusDisconnecting:
        end_ringing = time.clock()
        print "::A call is being disconnected"
        print end_ringing

    elif newState == telephone.EStatusIdle:
```

```
            print "::Idle"
            e32.ao_sleep(1,simar)

def simar():
   print "\n\n Inside simar"
   print end_ringing
   print start_ringing
   diff = (end_ringing-start_ringing)
   print diff

   if diff<2:
            print "\n:::::0 is sent::::::"
            appuifw.note(u":::::0 is sent::::::", "info")
   elif diff<5:
            print "\n:::::3 is sent::::::"
            appuifw.note(u":::::3 is sent::::::", "info")
   elif diff<8:
            print "\n:::::6 is sent::::::"
            appuifw.note(u":::::6 is sent::::::", "info")
   elif diff<11:
            print "\n:::::9 is sent::::::"
            appuifw.note(u":::::9 is sent::::::", "info")
   elif diff<14:
            print "\n:::::12 is sent::::::"
            appuifw.note(u":::::12 is sent::::::", "info")
   elif diff<17:
            print "\n:::::15 is sent::::::"
            appuifw.note(u":::::15 is sent::::::", "info")
   elif diff<20:
            print "\n:::::18 is sent::::::"
            appuifw.note(u":::::18 is sent::::::", "info")
   elif diff<23:
            print "\n:::::21 is sent::::::"
            appuifw.note(u":::::21 is sent::::::", "info")
   elif diff<26:
            print "\n:::::24 is sent::::::"
            appuifw.note(u":::::24 is sent::::::", "info")
   elif diff<29:
            print "\n:::::27 is sent::::::"
            appuifw.note(u":::::27 is sent::::::", "info")
   elif diff<32:
            print "\n:::::30 is sent::::::"
            appuifw.note(u":::::30 is sent::::::", "info")
   elif diff<35:
            print "\n:::::33 is sent::::::"
            appuifw.note(u":::::33 is sent::::::", "info")
   elif diff<38:
            print "\n:::::36 is sent::::::"
            appuifw.note(u":::::36 is sent::::::", "info")
   elif diff<41:
            print "\n:::::39 is sent::::::"
            appuifw.note(u":::::39 is sent::::::", "info")
   elif diff<44:
            print "\n:::::42 is sent::::::"
            appuifw.note(u"::::42 is sent::::::", "info")

telephone.incoming_call()
telephone.call_state(newphonestate)
```

Sender – This code sends a time restricted miss call to the number provided in the code.

```
import telephone
import e32
import appuifw


number = 9915326619
```

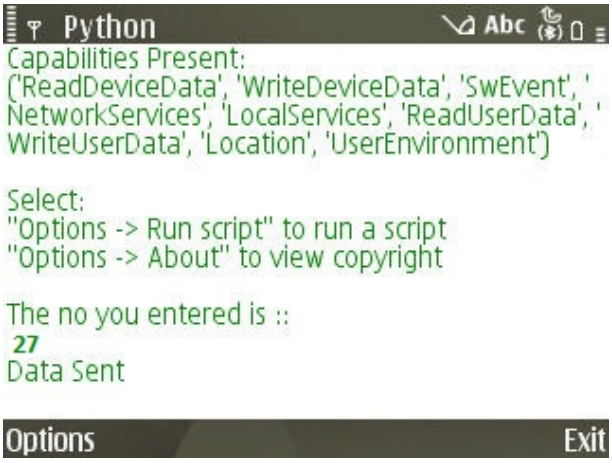
```
#There will be a pop up with a input box that asks for 'call time'
duration = appuifw.query(u"Enter the Durations",'number')

telephone.dial(str(number)) #make call

#define the callback handler function
def handle_hang_up(status):
   if status[0] == telephone.EStatusConnecting:
              print "\nConnecting"
      e32.ao_sleep(float(duration), telephone.hang_up)

telephone.call_state(handle_hang_up) #set the handler function
```

Now we will send only 15 values from sender instead of all and see whether they are transferred to the receiver or not.

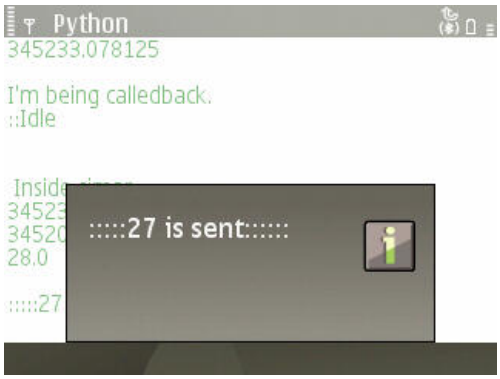
How things go

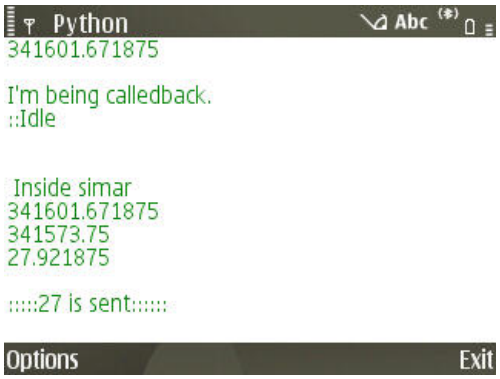
Now I am depicting, only the changes that I have made in the software. Just to avoid any confusion, I sent a 27 'call time' from the sender. Here is what we get in receiver.

Sender



Receiver





Logs

Here I’m some logs of outputs I got while testing.

Receiver

```
I'm being calledback.
::The new phone is ringing, call is from +918146010177
341483.796875

I'm being calledback.
::A call is being disconnected
341485.109375

I'm being calledback.
::Idle


 Inside simar
341485.109375
341483.796875
1.3125

:::::0 is sent::::::

I'm being calledback.
::The new phone is ringing, call is from +918146010177
341505.0

I'm being calledback.
::A call is being disconnected
341511.921875

I'm being calledback.
::Idle


 Inside simar
341511.921875
341505.0
6.921875

:::::6 is sent::::::

I'm being calledback.
::The new phone is ringing, call is from +918146010177
341535.28125

I'm being calledback.
::A call is being disconnected
341545.296875

I'm being calledback.
::Idle


 Inside simar
341545.296875
341535.28125
10.015625

:::::9 is sent::::::
```

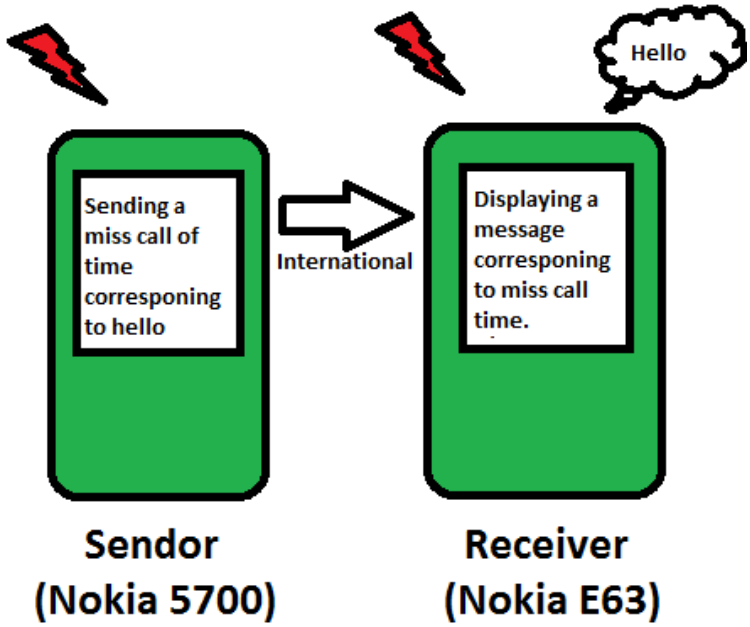
Sender

```
The no you entered is ::
0
Transferring Data ....


Data Sent
The no you entered is ::
6
Transferring Data ....


Data Sent
The no you entered is ::
9
Transferring Data ....
```

What Stage TWO leads to

I did not get even a single error in message transfer up till now. So this algorithm was ready to be put into a nice user friendly GUI and converted to useful applications. With the success of Stage 2 software I got a feeling of satisfaction.

Stage THREE – 1/9/2010



As I have done a lot of testing in previous stages, now I plan to focus more on applications of this technique. So in all further stages I have made application based software.

In this stage I have put the same software of stage 2, with slight modifications, in a nice GUI frame. Stage 3 introduces application based software that has template messages which we can send across the world for free. Yes even abroad so now you can avoid costly international SMS and calls wherever possible.

Receiver - This code calculates the time of miss call, interpret the 'time difference' and display the message, or plays music according to what the sender wants to do.

```
import telephone
import time
import e32
import appuifw
import audio

song = audio.Sound.open(u"e:\Sounds\Digital\Jazzy B- Romeo.MP3")

def newphonestate (stateInformation):

    global start_ringing
    global end_ringing

    newState = stateInformation[0]

    if newState == telephone.EStatusRinging:
        start_ringing = time.clock()

    elif newState == telephone.EStatusDisconnecting:
        end_ringing = time.clock()

    elif newState == telephone.EStatusIdle:
            e32.ao_sleep(1,simar)

def simar():
```

```
    diff = (end_ringing-start_ringing)
    print diff

    if diff<2:
            # Displays message in the form of a popup.
            appuifw.note(u"hello", "info")
    elif diff<5:
            appuifw.note(u"listen to a song.. wait", "info")
            # This plays a song on the receiver's phone..
            song.play(1)
    elif diff<8:
            appuifw.note(u"love  you", "info")
    elif diff<11:
            appuifw.note(u"i'm busy.. cya later", "info")
    elif diff<14:
            appuifw.note(u"i will call you later", "info")
    elif diff<17:
            appuifw.note(u"cheers!!", "info")
    elif diff<20:
            appuifw.note(u"how are you?", "info")
    elif diff<23:
            appuifw.note(u"absolutely fine", "info")
    elif diff<26:
            appuifw.note(u"have a good day", "info")
    elif diff<29:
            appuifw.note(u"regards", "info")
    elif diff<32:
            appuifw.note(u"i will c you later", "info")
    elif diff<35:
            appuifw.note(u"i'm free", "info")
    elif diff<38:
            appuifw.note(u"i'm sleeping", "info")
    elif diff<41:
            appuifw.note(u"hi", "info")
    elif diff<44:
            appuifw.note(u"shut up please.", "info")

telephone.incoming_call()
telephone.call_state(newphonestate)
```

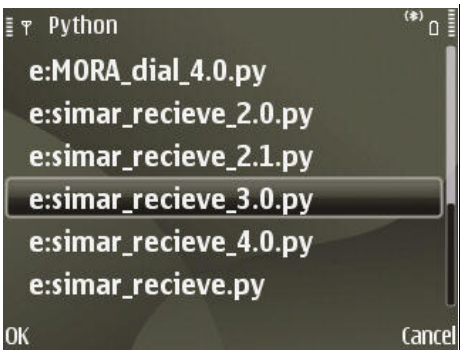
Sender – This code presents a very nice format of GUI to send template messages to the receiver.

```
import telephone
import e32
import appuifw

number = 9915326619 #Enter the number you want to call

appuifw.app.screen='normal'

# Graphics related configration
txt = appuifw.Text()
appuifw.app.body = txt
txt.color = 0x3A5FCD
txt.font = u"LatinBold25"
#txt.highlight_color = 0xc0c0c0
txt.style = appuifw.STYLE_BOLD
txt.style = appuifw.HIGHLIGHT_SHADOW
txt.add(u"SIMAR\n")

def loop():
   global duration       #global variable
   M = [u"Hello",u"Play a song",u"Love you",u"I'm busy .. c ya later",u"I

will call you later",u"cheers..",u"how are you",u"fine absolutely",u"have a

good day",u"regards",u"i will c you later",u"i'm free ..",u"i'm

sleeping",u"hi",u"Shup up"]
```

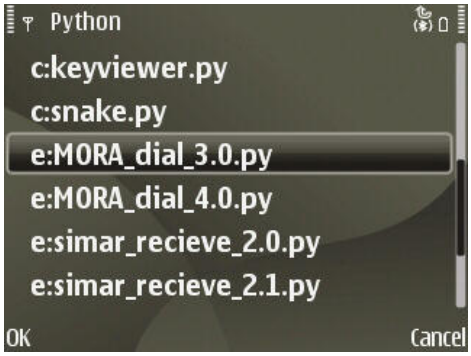
```
    # gives the index of the selection made in the list..
    index= appuifw.selection_list(choices=M, search_field=1)

   if index==0:
       duration = 0
    elif index==1:
       duration = 3
    elif index==2:
       duration = 6
    elif index==3:
       duration = 9
    elif index==4:
       duration = 12
    elif index==5:
       duration = 15
    elif index==6:
       duration = 18
    elif index==7:
       duration = 21
    elif index==8:
       duration = 24
    elif index==9:
       duration = 27
    elif index==10:
       duration = 30
    elif index==11:
       duration = 33
    elif index==12:
       duration = 36
    elif index==13:
       duration = 39
    elif index==14:
       duration = 42

    telephone.dial(str(number)) #make call

#define the handler function
def handle_hang_up(status):
    #test if the call was complete

    if status[0] == telephone.EStatusConnecting:
        # Makes the application sleep for 'duration' seconds and then call the function telephone_hangup
        e32.ao_sleep(float(duration), telephone.hang_up)

    elif status[0] == telephone.EStatusDisconnecting:
                e32.ao_sleep(1, loop)


telephone.call_state(handle_hang_up) #set the handler function

loop()

#appuifw.app.exit_key_handler=quit
#app_lock=e32.Ao_lock()
#app_lock.wait()
```

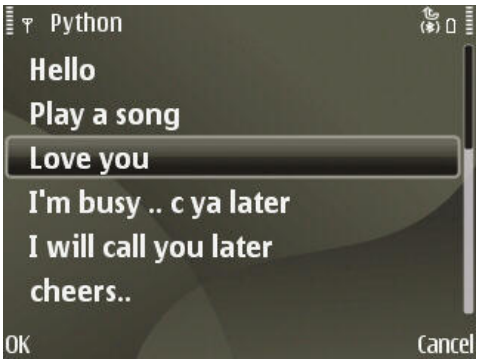
How Things Go – I'm providing you the snapshots of the events how the software works

Receiver – software must be running on receiver

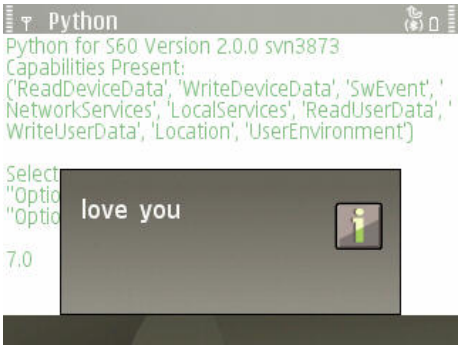


Sender – With the software running on receiver use sender to send messages many times





Receiver



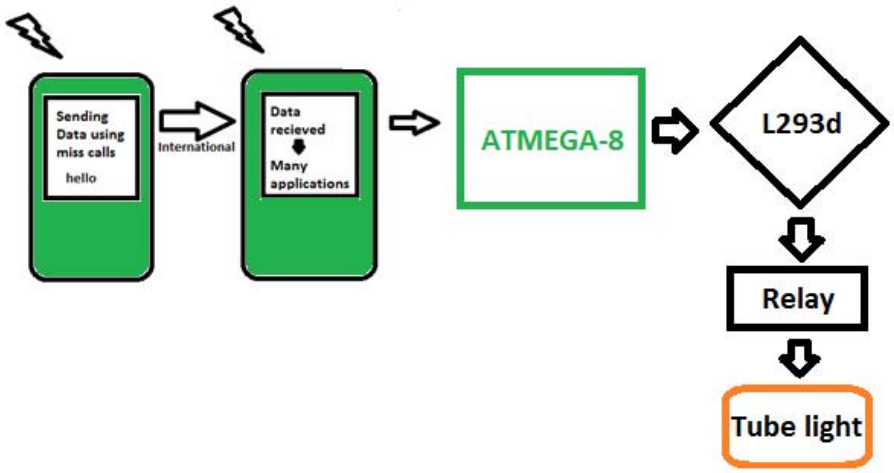
Practical Importance of the software

I plan to release a standalone copy of this application as I got a very good response from my friend circle. This way I will be able to test it well and feedback will help me to improve its implementation. However the main improvement to focus as of now, is the introduction of two way communication, so users will be able to send a reply using the same software.

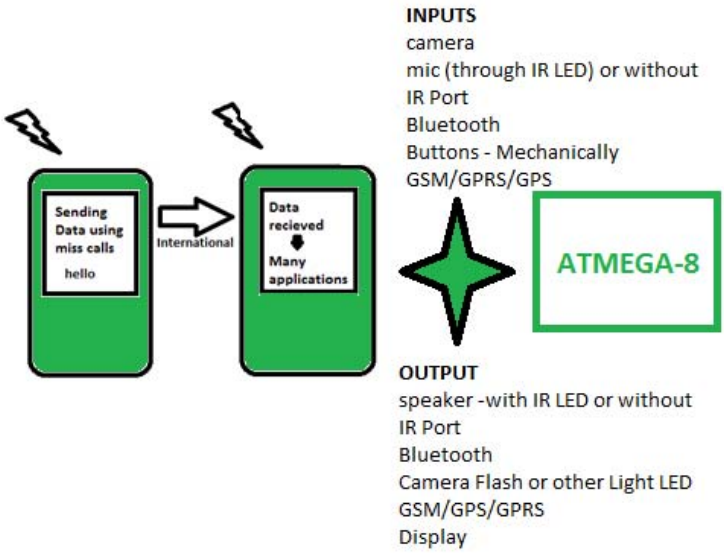
Another thing that fascinates people is playing a song when a command is sent from a sender I have already implemented it in the Stage 3 software and it has amused many people. An example of its implementation is included in the video in the CD, accompanied with the project report.

Besides this I plan to modify this software to create some other similar innovative applications like exchanging smilies with friends, displaying pre-recorded video messages in your loved ones mobile. Besides, I know though that it will not be a practically successful endeavour but still I will make the software for general purpose message transfer that will be just like SMS but free, globally. Its only limitation will be that multiple miss calls will have to be sent to transfer the complete message.

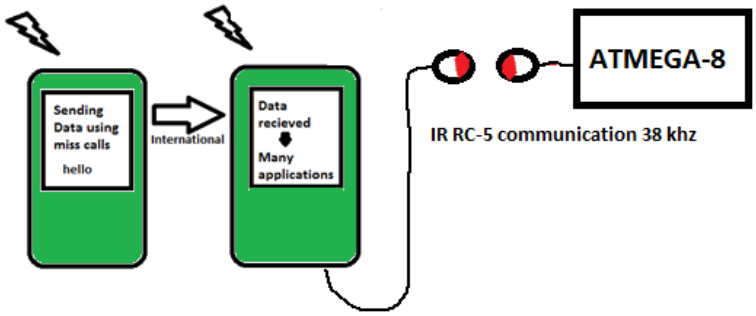
Stage FOUR – 4/9/2010



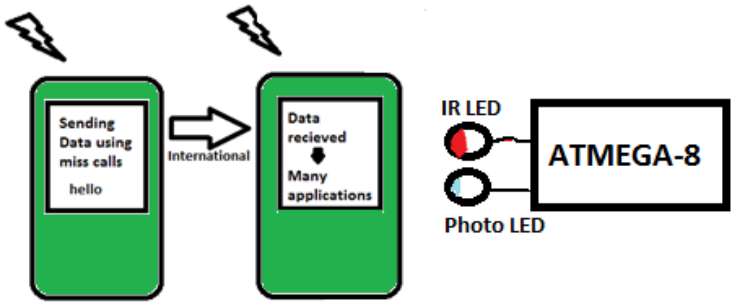
Stage 4 software is also application based software. It forms the basis of a vast group of further applications possible by the transferring the information that receiver receives to external world through microcontrollers. But, how to interface a mobile with the microcontroller is another question. I tried searching/asking in forums for simplest implementation. The only solution I got from there is Infra Red. So, I started working on IR which I thought was the simplest approach. But, a desire to work on something different, simple, easy and innovative alternative approach forced me to consider brainstorming the possible ways or 'ports' from which mobile phone can interact with the outside world.



Out of these I was particularly focussed on output from a mobile as of now. The very first innovative approach that I thought was the use of audio jack as it can produce signal, a time varying signal. Unfortunately searching on the net I found that there already exist an well implemented and patented solution of using an IR led with the audio jack and its implementation was not really easy but rather complex.



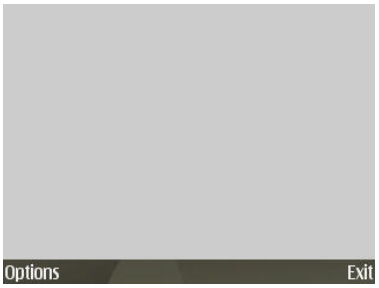
The next thing that struck in my mind was display. Isn't there a way that the microcontroller can sense the display and get meaningful information out of it just as we human beings do using our eyes? The moment this thought came to my mind, I thought of the eyes of microcontrollers, ah!, its IR sensor, that is typically used as an obstacle detector and in line detection in robots.



I have to use a black, white screen and many light shaded screens to get decreasing voltages that can be fed to the A/D Convertor of a microcontroller to communicate. Another main benefit of using this technique to interface with MCU is that the two subsystems are completely independent of each other.

0x000000 0x222222 0x666666

0xaaaaaa



0xcccccc

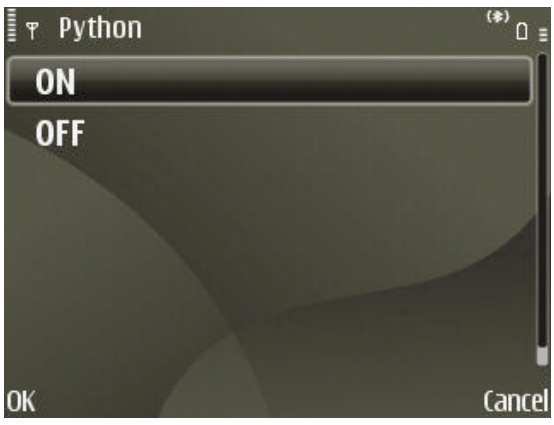
0xffffff

We can get six values of voltages from IR sensor so six informations can be transmitted to the MCU. However due to shortage of time I have implemented just an ON/OFF Black/White application using AVR ATmega8 and a relay interfaced to LM293d IC.

How Things Go – I'm providing you the snapshots of the events how the software works

Receiver – software must be running on receiver. It is initially in OFF state.

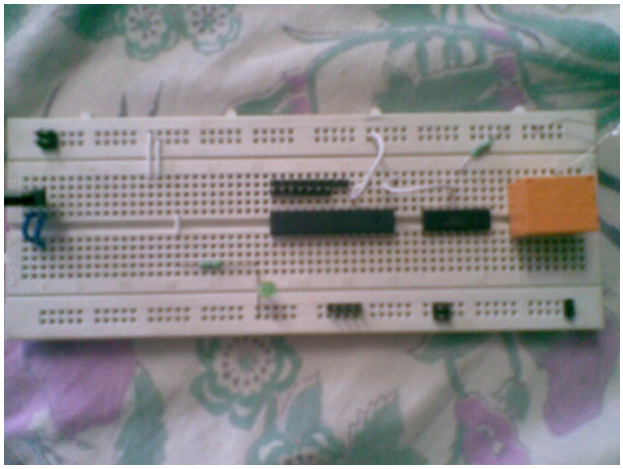
Sender – Turn on the tube light



Receiver

The black signal produced a 5V i.e 1 logic at the PORTB pin0 is Atmega8 which we detected using simple polling system to set the PORTD pin0, connected to relay using LM298 IC. So we can change the state of really and hence the tube light using ON/OFF from sender. Now have a look on the software code.

AVR



Receiver

```
import telephone
import time
import e32
import appuifw

# Makes the application remain and not auto quit..
# This is essential code that has to be included if making a canvas application
def quit():
   App_lock.signal()

appuifw.app.exit_key_handler = quit

appuifw.app.screen='full'

canvas=appuifw.Canvas()
appuifw.app.body=canvas

## Clear the canvas with given color
canvas.clear(0xffffff)                  #rgb oxrrggbb
```

```
def newphonestate (stateInformation):

   global start_ringing
   global end_ringing

   newState = stateInformation[0]

   if newState == telephone.EStatusRinging:
      start_ringing = time.clock()

   elif newState == telephone.EStatusDisconnecting:
      end_ringing = time.clock()

   elif newState == telephone.EStatusIdle:
            e32.ao_sleep(1,simar)

def simar():
   diff = (end_ringing-start_ringing)

   if diff<2:
            # fills the whole screen with the given color..
            # the argument is of the form 0xrrggbb
            canvas.clear(0x000000)          #black
   elif diff<8:
            canvas.clear(0xffffff)          #white


telephone.incoming_call()
telephone.call_state(newphonestate)

App_lock = e32.Ao_lock()
App_lock.wait()
```

Sender

```
import telephone
import e32
import appuifw

#def quit():
#    App_lock.signal()
#appuifw.app.exit_key_handler = quit

##app code starts
number = 9915326619 #Enter the number you want to call

appuifw.app.screen='normal'

txt = appuifw.Text()
appuifw.app.body = txt
txt.color = 0x3A5FCD
txt.font = u"LatinBold25"
#txt.highlight_color = 0xc0c0c0
txt.style = appuifw.STYLE_BOLD
txt.style = appuifw.HIGHLIGHT_SHADOW
txt.add(u"SIMAR\n")

def loop():
   global duration
   M = [u"ON",u"OFF"]
   index= appuifw.selection_list(choices=M, search_field=1)
   if index==0:
      duration = 0

   if index==1:
      duration = 6
   telephone.dial(str(number)) #make call
```

```
#define the handler function

def handle_hang_up(status):
   #test if the call was complete

   if status[0] == telephone.EStatusConnecting:
      e32.ao_sleep(float(duration), telephone.hang_up)

   elif status[0] == telephone.EStatusDisconnecting:
            e32.ao_sleep(1, loop)


telephone.call_state(handle_hang_up) #set the handler function

loop()

#appuifw.app.exit_key_handler=quit
#app_lock=e32.Ao_lock()
#app_lock.wait()
```

AVR – ATMega8

```
#define F_CPU 12000000UL
#include<avr/io.h>
#include<util/delay.h>

#define INPUT 0x00
#define OUTPUT 0xFF

int main()
{

   //Setting up the data direction registers
    DDRD= OUTPUT;
    DDRB= INPUT;

    while (1)
            {
                    if (PINB & (1<<0))
                    {
                            PORTD = 0b00000001;             //AC ON
                    }
                    else
                    {
                            PORTD = 0b00000000;             //AC OFF
                    }
                    _delay_ms(150);
            }

}
```

Improvements

I have shown above how we can transfer more than just ON/OFF from mobile phone to microcontroller so in a way, if we can transmit 15 informations, we can control more than just one device, 7 devices in all with just one miss call. Moreover, putting it in a little different way, not just we can an AC ON/OFF but also control its temperature. Now this is interesting, we can put an AC ON/OFF and also control the temperature to 13 different values while still in drive to home.

Summing up

Further improvements

Till the last day I posted the project report, I had been working on different applications and their improvements. Stage 4 was started as late as 4/9/2010 and there is still a big open window open to wander and make interesting applications that will surely find a place into our everyday life.

I have already mentioned the immediate improvements possible, regarding stage-3 and stage-4 software in the respective blocks of the report itself. However I would like to emphasize one point here that I seemed to forget after I achieved required precision in stage-2.

We have sacrificed our resolution for the sake of precision. Surely while improving and generating more applications, we will feel the need of more resolution over the 45s miss call range, which will enable us to transfer more information.

It seems that the resolution can be worked out to a great extend looking at the data I recorded in stage-2. However I skipped a very interesting observation there which I will like to add here now.

"It is interesting to note the big leap in values corresponding to 3s when I just put the phone on silent mode from ringing mode. Yes, just putting my receiver (Nokia E63) on silent mode and a huge change in values. This means that there is a precision loss right inside the device and not in network. After tweaking a bit with time.clock() and time.time(), I found the culprit is none other but python. Its either the the callbacks are precise or the clock() itself. Now how to figure this out is still Greek to me. I have already used the most precise and accurate clock() available in pyS60. So, whether the clock is giving non-precise values, there is overhead in callbacks or a precision loss in network still remains unresolved."

So this is an important improvement that remains, so I decided to implement this entire in C in future rather than in python. I fell happy to share that I have already worked in this regard and have successfully run a 'Hello World' program in my Nokia 6600.

Also no protocol is error free whether it is Bluetooth or wireless all has errors, so to tackle these they have EDAC (Error Detection and Correction). It depends on the amounts of errors that what will be the transferring time. Techniques such as Bit parity check and checksums can be applied in this project also to tackle with errors and increase resolution.

I hope to achieve more precision after I implement the project in C, to the extent that at least 27 values can be distinguished precisely. So, this way we can send an English alphabet in just one miss call. This will help me a lot in transferring a whole line text using multiple miss calls.

I think there are a lot of improvements possible in the project to increase resolution and increasing the speed of information transfer.

How far I think, I have reached.

I would like to add here one thing that it should not be compared to SMS. Surely it is no alternative to SMS. I have not used any frequency or available technologies that a mobile service provider provides nor do I have any knowledge of it. I have just had an innovative idea and found a way in which I can transfer some data keeping the details of call in a black box. So it has certain limitations.

Moreover, I don't know whether it is legal to use such a thing or not but all these things apart it has its own importance in free remote appliance control, sending free international template messages and in many many other forms. I hope this idea can easily replace remote control applications using a mobile phone such as SMS control. Making international remote application control had never be so easy. I took the response of my friends, they like it very much. Moreover Dr AK Verma, my supervisor also praised the idea very much.

Conclusions

Another thing I would like to mention here that although we have found a big loop hole in python and limitation of using it is quite clear now, it's not like python is a good language and we shall avoid it rather I think python has really paid off for my project.

Without python may be I had not been able to implement the idea at all. Note here that this is the biggest benefit of python, you can concentrate right on the idea as the implementation in python is very easy. Once done you can port your project to whichever language suits your project the best. So hats off to python.

I think I have found a very essential tool that will help me (at least) in my all projects from now onwards. So in this regard python has a very important language and it has its own unique place in programming world that cannot be replaced by any languages like C. So we cannot compare python with C.

Besides the technical aspects, I feel this is one of most important thing that I have learnt while implementing this project.

References

1. http://www.symbian.org/
2. http://en.wikipedia.org/wiki/Symbian_OS
3. http://wiki.opensource.nokia.com/projects/PyS60
4. http://developer.symbian.org/wiki/index.php/Python_Quick_Start
5. https://garage.maemo.org/frs/?group_id=854
6. https://www.avrfreaks.com
7. pyS60 official documentation (included in CD)