Made from the 16/7 to the 23/7

EPQ:

**The project:**

The base of this idea is that I want to be able to control a remote control car remotely. I then had [x] decisions to make:

Simple over view:

1. What hardware to use on the remote control car: modify the existing remote, a small pc (raspberry pi) or a mobile phone.
2. How the car will communicate with the controller: Bluetooth, radio waves, internet or infrared
3. What hone to use: android, windows, IOS, Ubuntu phone…
4. The coding language to use: Java, python, C# or C++

Detailed overview and conclutions

1. Modify existing remote: This would require me to take apart the existing remote and modify it such that a pice of hardware could control and create contacts that would normaily be done by pressing buttons on the remote. This would require something with GPIO or out put pins to control the remote with. But this would make unnessasery bulk on the car or a “base station” that the car would communicate with. This would lightly be easy to make and modify, but would require extra hardware and possibly extra cost.   
   Use a small pc/raspberry pi(exclusively): this would require me to take apart the remote control car and modify the internal electronics so that I can use the GPIO pins and some simple electronics( mosfets or transistres) to turn the motors on when they are required. This may required me to have knolage of the internal electronics of a the car, I may be able to get this by using the existing remote and using a voltmeter and a coustom battery pack ot power them. This could be hard to do as I will lightly nt be able to get this info from the manufacture as it will lightly be a company secret.   
   Using a mobile phone: this would be small, light and I may have a spare android phone. But I would need to find a way of controlling the car with it, if the car uses Bluetooth then I could potentially communicate with it from the bluetoth on the phone. But that would be near imposible.   
     
   Conclution:  
   I will try to use a small pc/raspberry pi because that is the most effective solution, but I may want to modify the existing remote if that is to hard.
2. Bluetooth: This is device to devices whitch means some things are simple and should not be to hard to program. But range is very limited and it can be easily hacked and the connection may not be secure. Also most smartphones have Bluetooth built in and there are USB atachments that are cheap to provide Bluetooth.  
   Radio Waves: This car be hard to use as other devices easily can interfere with it and it requires a module that is not small, furthermore I would need to deside on a frequency and get 2 radio modules that could be expencive and hard to control with out external liberys that could be complicated.   
   Infra red: I can use usb adapters that are cheap, but they cant be used in sunlight and often require dicret line of sight, hard on a moving vehicle. This is also not easy in most languages as infra red has largely been replaced by blue tooth.   
   Internet: This is very well documented in most languages, easy to use and this would menan it can be acsesed almost everywhere. This could increase latency thow and would require a coustom protocol, however this would be easy to do and simplify the process of sending and reciving data. Also if I use different language on the car and the reciver then I may have to deal with byte code whitch can get complicated.

Conclution: if I can use the internet that would be good due to its versatility and that it is everywhere. But I may resort to Bluetooth if I am using 2 different languages.

1. Android: This can be programed in Java or C/C++. Java is more documented and often considered easyer to program. Java has a built in internet libery that is well documented and easy to use, java can be run on a raspberry pi but the performance is not very good on the pi (it is near native on on android using the new ART runtime) however because there will only be 1 connection at a time there is no need to worry about performance as it will not be doing a lot. C/C++ is hard to program and has a more complicated libery system and the proformance is[on both sytems]. But it is not very well documented.  
   IOS: This is programed primarily in objective C, to use this however you require a mac and a £99 a year licence I don’t have that money.   
   Windows Phone: I am unfamilyer with windows phone and how to program it and tere for the required liberys but it has little adoption and a very small market share.   
   UPDATE[date]:  
     
   Ubuntu phone: This is being lanched as a very new OS and adoption is very low and it only works on some phones. HTML5 is the language being pushed for the platform, this means it would work as a web applet with little adaptation.

Conclution: Android is the OS I will target this at because of its wide adoption and the fact that it is fere to develop for and easy and well documented.

1. Java: this will be good to use because java is my language of choise for android there for I don’t have to deal with byte code furthermore it has a well-documented built in libery this means that I don’t have to worry about licences for other libery or bad documentation. But it dose not have a inbuilt libery to use the GPIO pins on a raspberry pi. But there is a libery called [] because of the nature of what it is doing is simple to use and the documentation is very through. I will have to lern how to import the libery and use it thow.  
   python: this is the language that is is being pushed on the raspberry pi, as a result there is a lot of documentation for it further more it is rather low level this means that the proformance of it is very good. However it relys on indentaion rather than bracket in its syantax this means that it can be hard to debug. However I am unaware if python requires and external libery to use the internet and how the byte code between this and java will work.  
   UPDATE: python comes with an inbuilt libery for networking.  
     
   C family(C, C++ or C#): for most of these they require an externeal libery for internet E.G. CURL this is well made and know to work but other that offital documentationit is hard to find help about it (E.G. tutorials) however I need to do more resuch on these languages to see how easy byte code is to do and to see if it can be used to control the GPIO pins

UPDATE: there is a libery for C and C++ called [wiring pi](http://wiringpi.com/) so I can use this.  
  
Conclution: I will use Java due to its wide adoption, the fact that I can eliminate the need to deal with byte code and its built in liberys.