# Idea

Write an embedded firmware to receive a packet of information via UART, process it

and send it back on the same UART.

The embedded firmware shall:

1. Accept packet of data via a UART at 9600 baud

2. Base64 encode it

3. Transmit the resultant data back in the same packet format



<data> shall represent an ASCII string stored as 8-bit unsigned characters

<data-len> represents the length of <data> stored as an MSB first 16-bit number

<8-bit-crc> represents an 8-bit CRC with the polynomial 0x85 of the entire packet

except the CRC byte (bytes 0 .. 2 + <data-len> inclusive)

# Design



# Scope

Setup a commincation channel over UART, between a Host (x86 PC) and a Target (Raspberry Pi 1, Model B). This communication channel is used to exchange command-response information.

# Detailed design

### Assumptions

* Target device is a Raspberry Pi 1, Model B.
* The data length is 2 ASCII charaters wide, that allows a data size of 255 characters.

### Components (software)

* Base64 encoder
* ASCII to length modifier

# Detailed design

### Baud Rate calculation

The IBRD and FBRD registers specify the baud rate. The baud rate divisor is calculated as follows:

* Baud rate divisor BAUDDIV = (FUARTCLK/(16\*Baud rate)) where FUARTCLK is the UART reference clock frequency.

Baud rate = (UART\_CLOCK)/(16\* (m+n/64), where m = interger part, n = fractional part

UART\_CLOCK = 3,000,000; Baud = 9,600.   
IBRD = int(3,000,000/(16\*9,600)) = int(19.53) = 19  
FBRD = round(0.53\*64) = 34

Why 64 ? FBRD is 6bits wide, muliplying the fractional part by 64 gets you closest to the REQUIRED baud rate. Its reduces error.

### Source code

<https://github.com/bhargav89/raspberry-pi/tree/master/bare_bones>

### Build requirements

* Raspberry Pi 1, Model B and an SD-card.
* USB to TTL (USB<->Serial conversion chip).
* Build instructions
  + Clone the repository into your local machine.
  + Run the following command chmod u+x build.sh
  + Insert the SD card into the PC and run ./build.sh flash
  + Plug in the SD card into the raspberry pi board, fire up a serial communication based program on the PC (Ex: Minicom, Kermit) (Reference image in Appendix A)
  + Transmit and receive packets to the Target using Minicom/Kermit.

# Limitations of the design

* The CRC computation is not handled.

# Appendix A

