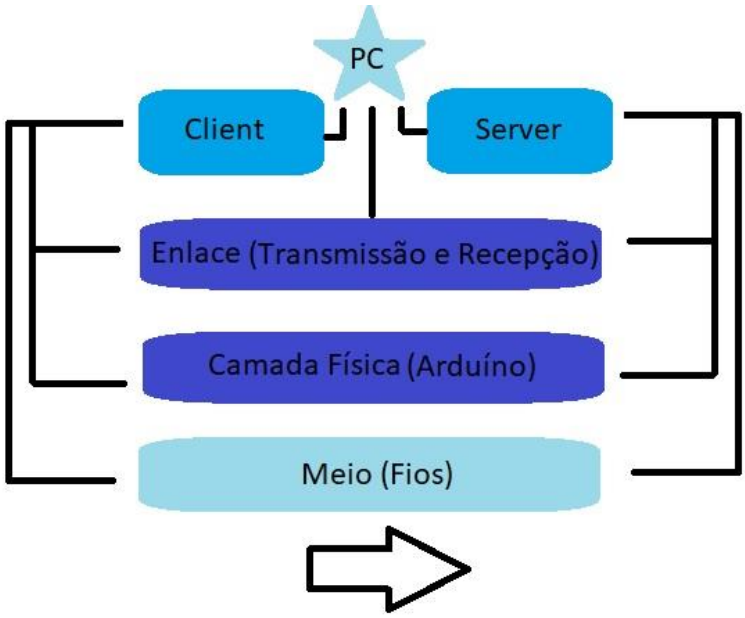
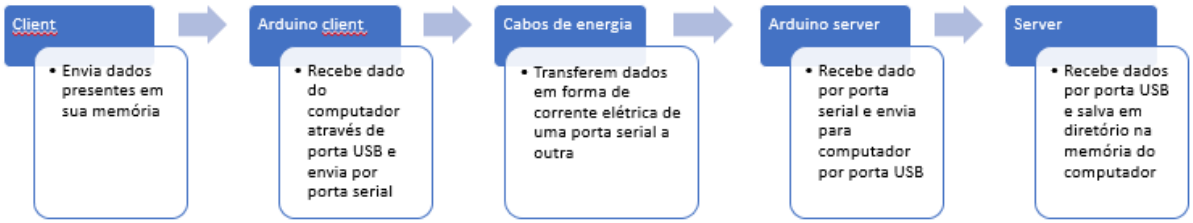
Projeto 1

Diagrama de funcionamento e camadas





Vídeo do funcionamento do sistema

https://youtu.be/180gFHwVKQc

Projeto 2

Desenho do pacote

Head		Payload	EoP
Start	Size	Conteudo	End
0xFF	len(data)	Data	0xFE

Dois bytes do Head guardam o tamanho, e um significa o começo.

O End of Packet não representa muito por enquanto, pois não foi usado em nosso método. Contudo, representa 1 byte do packet e será usado futuramente.

Cálculo do Overhead

$$OverHead = \frac{TamanhoTotal}{TamanhoPayload}$$

O Overhead é a razão entre o tamanho total e a carga útil. Neste caso, é payload+4/payload. Ou seja, 1+(4/tamanho do payload).

Para um payload de tamanho 3000, esse valor da 1.00133

Cálculo do tempo teórico de transmissao de qualquer imagem

A velocidade padrão de transmissão pela porta serial é de 9600b/s. Multiplica-se por 8 para obter os bits, e soma-se o valor do head+eop (4*8).

$$Tempo\ de\ Transmissao = \frac{9600}{Imagem * 8 + 32}$$

Cálculo do Thoughput

$$Troughput = Tamanho\ do\ Payload + 4$$

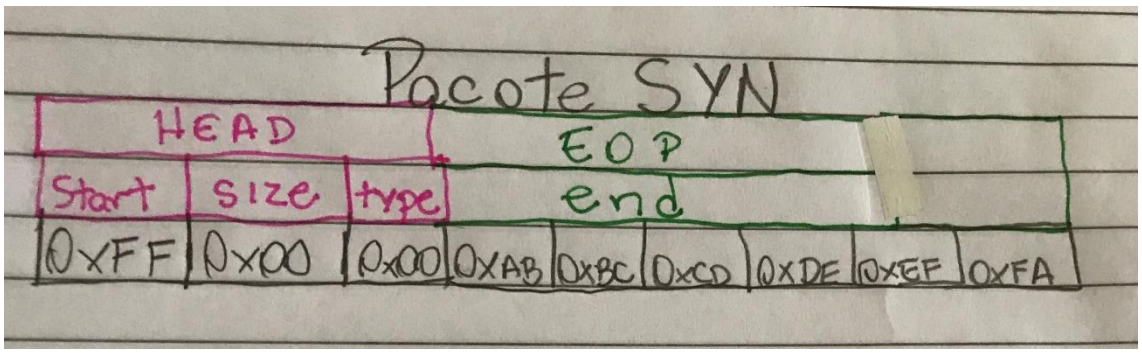
O tamanho da imagem somado aos valores de bytes do Head e do EOP fazem o Troughput. No caso, tamanho + 4.

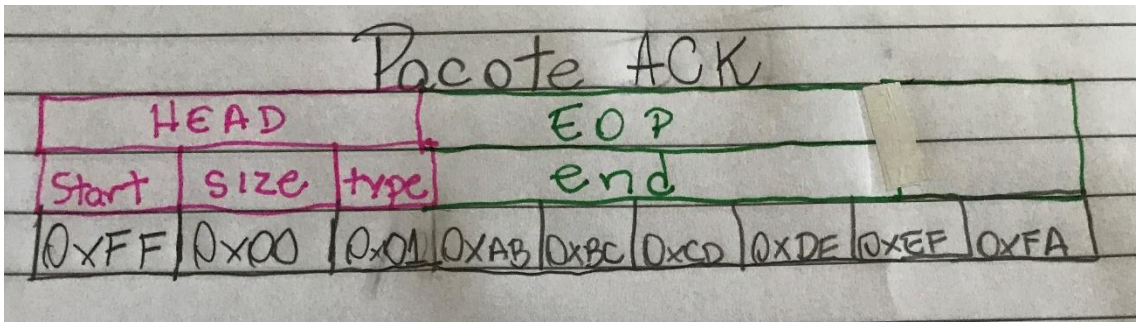
Projeto 3

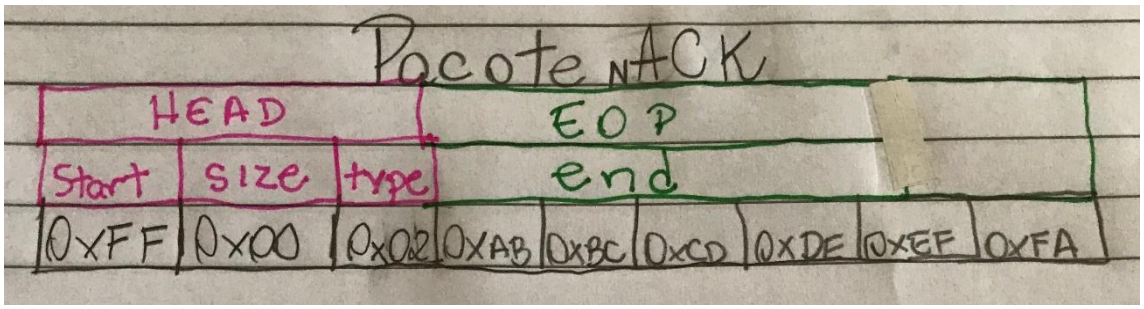
Descrição do HandShake implementado

O client envia um comando do tipo SYN e quando o server recebe, o server responde com dois comandos, um ACK e um SYN. Se o cliente passar 2 segundos sem ter recebido esses comandos, ele manda outro SYN e assim continua até que os receba. Quando ele recebe esses dois comandos o client manda um ACK e se o server recebe o ACK a comunicação está feita.

Descrever os pacotes (SYN, ACK, NACK e dados)







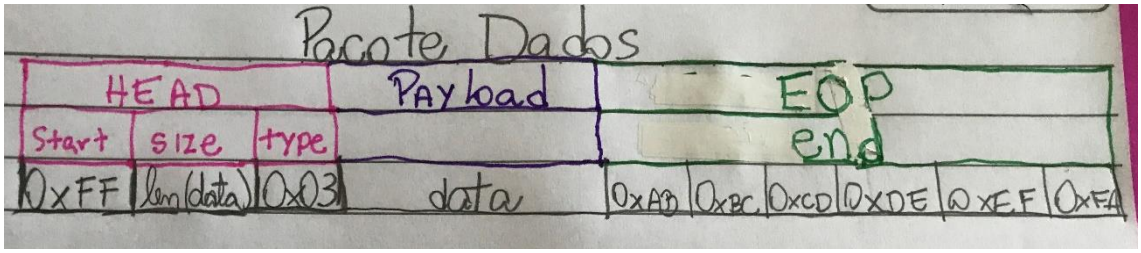


Diagrama do envio de pacotes como uma máquina de estados

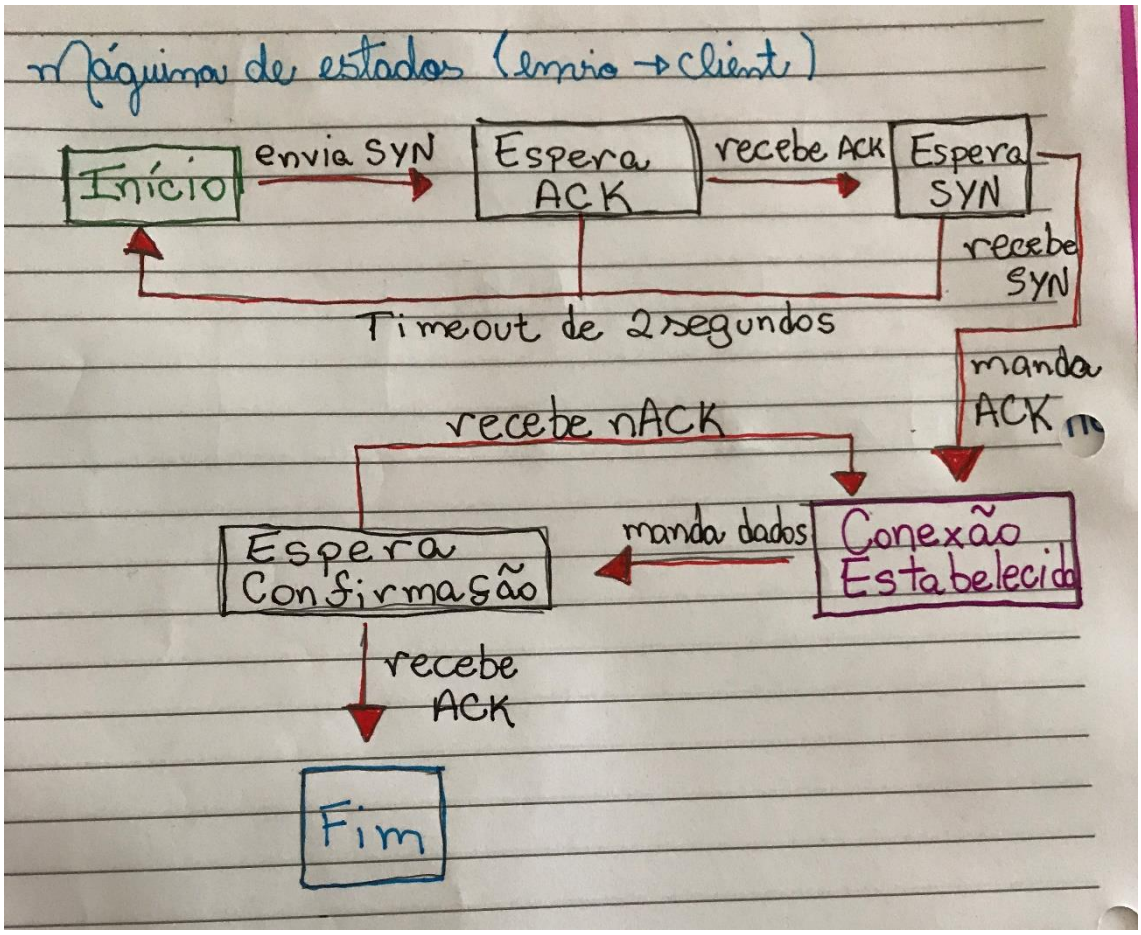
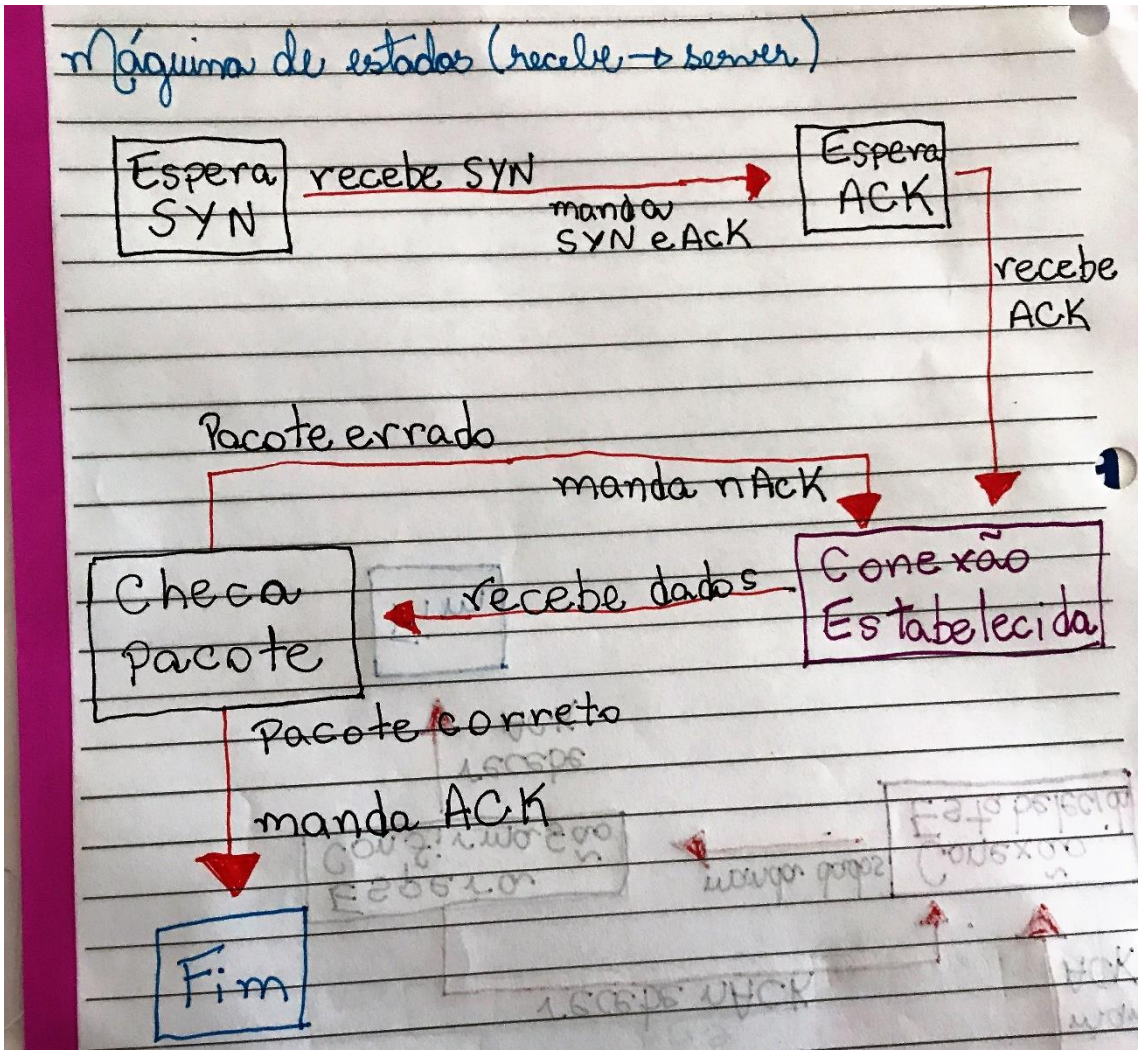


Diagrama a recepção de pacotes como uma máquina de estados



Descrever o tempo de timeout utilizado (e o porque desse valor)

O timeout utilizado foi de 2 segundos, a melhor otimização encontrada pela dupla ao perceber a recepção dos dados através da observação dos prints do recebimento.

Diferença entre pacotes de comando (SYN,ACK,NACK) de pacote de dados

Os pacotes possuem um byte reservado no HEAD que indica seu tipo (dados, SYN, ACK ou NACK) através dos números 0x03, 0x00, 0x01 e 0x02 respectivamente. Como os pacotes de comando SYN, ACK e NACK não possuem payload, seu size é 0x00.

Projeto 4

Fragmentação

O payload é dividido em pacotes de 2048 bytes, que são enviados em sequencia e recebem um ACK se chegaram completos ou um NACK caso haja algo errado. Se ele receber um NACK, ou não receber resposta por um timeout de 2 segundos, o pacote é reenviado. O timeout foi escolhido por parecer mais que suficiente para o grupo, após experiencia, para que haja a analise do pacote e envio do ACK se for o caso.

CRC

CRC, ou Cyclic Redundancy Check, é um algoritmo que analisa os dados como se fossem os coeficientes de um polinômio, e faz a divisão binária deles por um polinômio definido previamente. O resto dessa divisão é o retornado pelo CRC.

Escolhemos o CRC-8, $x^8 + x^7 + x^6 + x^4 + x^2 + 1$, por considerá-lo suficiente para os dados que estamos tratando, mantendo o Overhead baixo, e utilizamos a biblioteca *crcmod* no python.

Depois de implementar o CRC, são calculados os CRCs do HEAD e do payload do pacote recebido, e tais valores são armazenados no final do HEAD, portanto, são adicionados dois novos campos ao nosso antigo HEAD, o campo do CRC do Head e o campo do CRC do payload.

Posteriormente, conferimos se o CRC calculado do HEAD no tempo atual confere com o CRC que foi enviado e armazenado no mesmo, e o mesmo procedimento é feito para o CRC do payload, para confirmarmos que o pacote chegou corretamente.