Sistema de Eficiência Energética para Ar-Condicionado
(Ian Downie 15204948)

Requerimentos

Funcionalidade: cada vez que a porta ou a janela de uma divisão se abre, o condicionador de ar, se estiver ligado, desliga automaticamente e fica desligado até que a porta e a janela estejam fechadas e um botão esteja acionado para recomeçar o ar-condicionado.

Interface com utilizador: além do botão para reativar o condicionador de ar, há um LED que assinala se a porta ou a janela estão abertas e só desliga quando as duas estiverem fechadas. Adicionalmente, serão desenvolvidos um interface de notebook e de telefone celular para que o utilizador possa baixar um log dos seguintes eventos:

1. todos os eventos em um determinado intervalo de datas;
2. o tempo total (em horas e minutos) que o sistema manteve o ar-condicionado alimentado, num determinado intervalo de datas.

Desempenho: o sistema deve estar disponível para o utilizador durante todo o período que o ar-condicionado está em uso e não apresentar falhas que possam introduzir desvios da funcionalidade descrita acima.

Não funcionais

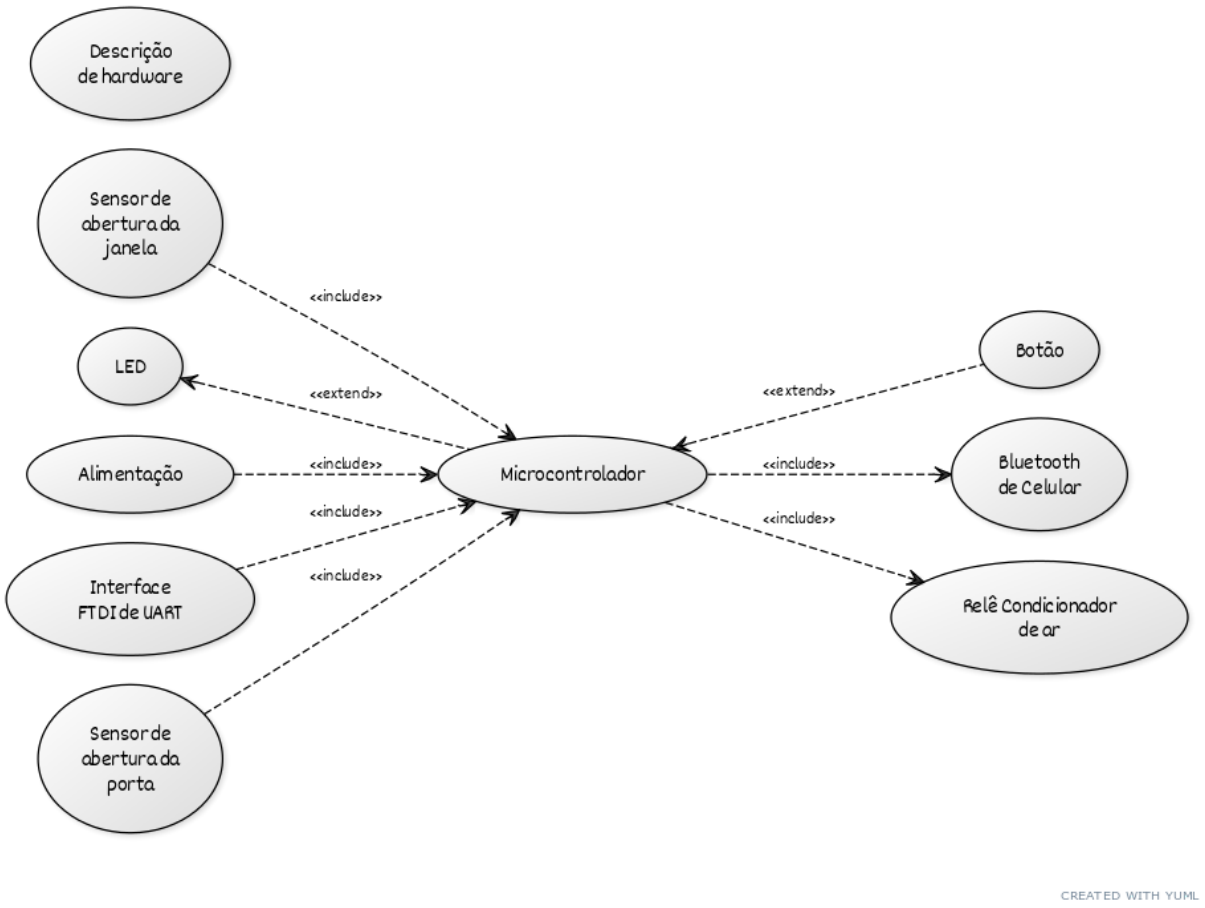
Custo: comparável com substitutos no mercado.

Tamanho físico e peso: O dispositivo não deve influenciar na projeção do resto do sistema de ar-condicionado, além de acomodar o LED.

Nome	**Sistema de Eficiência Energética para Ar Condicionado**
Propósito	Reduzir consumo de energia
Entradas	Botão para reativar o condicionador de ar, sensores de abertura da janela e da porta
Saídas	LED que assinala se uma porta ou uma janela estiver aberta; Interface UART para notebook; Bluetooth para celular.
Funções	Capaz de desligar logo que uma porta ou uma janela estiver aberta Capaz de ligar o ar-condicionado novamente imediatamente depois de um botão ser acionado Capaz de fornecer uma lista de eventos para um notebook ou telefone celular Capaz de organizar esta lista de eventos, através de software instalado no notebook ou telefone celular das duas seguintes formas: 1. todos os eventos em um determinado intervalo de datas; 2. o tempo total (em horas e minutos) que o sistema manteve o ar-condicionado alimentado, num determinado intervalo de datas.
Desempenho	Robusto sem necessidade de ser de alta velocidade
Não funcionais	
Custo de fabricação	Comparável com substitutos
Potência	40mW (window sensor 1,7mW; door sensor 1,7mW; ATmega48A, 0,36mW; LED, 36mW)
Tamanho	2cm*2cm*2 cm, mais espaço para acomodar sensores nas quadras da janela e da porta

Especificação

De acordo com os requerimentos do sistema, podemos descrever o sistema embarcado e a sua integração com o seguinte diagrama do hardware:



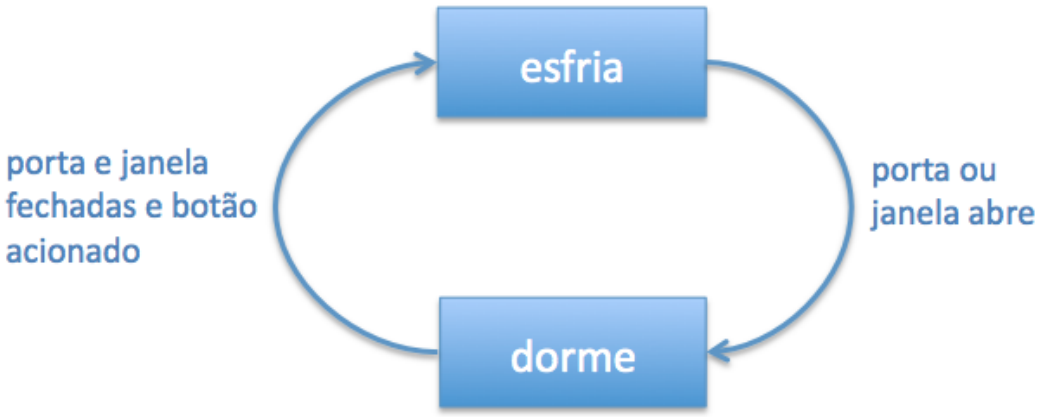
Podemos ver que o microcontrolador comunica com o utilizador através do LED e o botão, que pode reconectar o sistema de resfriamento se a porta e janela estiverem fechadas. Em adição a esta comunicação direta, o utilizador consegue acessar dados do sistema através da interface FTDI, que tem uma entrada de mini-USB para extrair os dados descritos nos requerimentos. Os mesmos dados são disponibilizados para um aplicativo no celular do utilizador via bluetooth. Por este motivo, o microcontrolador precisa de um módulo Wifi.

Embora seja usado um Raspberry Pico Pi na prototipagem do sistema, seria mais vantajoso em termos de custo substituir este kit de desenvolvimento por um microcontrolador que seja mais barato para produção em massa. Na especificação de potência, consideramos um ATmega48A.

Se escolhermos esta opção, temos a possibilidade de implementar as interrupções através de threads, em vez de utilizar os dois cores do Pico Pi.

Outra opção é utilizar o RP2040, o circuito integrado de microcontrolador de ARM Cortex-MO+ de dois cores que é utilizado no Raspberry Pico Pi. Na prototipagem com este circuito, utilizamos as biblioteca de hardware interrupções (irq) e de pinos gerais (gpio). Também, foi usado a biblioteca de funções de alto nível multicore. Adicionalmente, três bibliotecas de C/C++, <iostream>, <string> e <unistd.h>, foram utilizadas para operações de entrada/saída, manipulação de strings e a introdução de atrasos respectivamente.

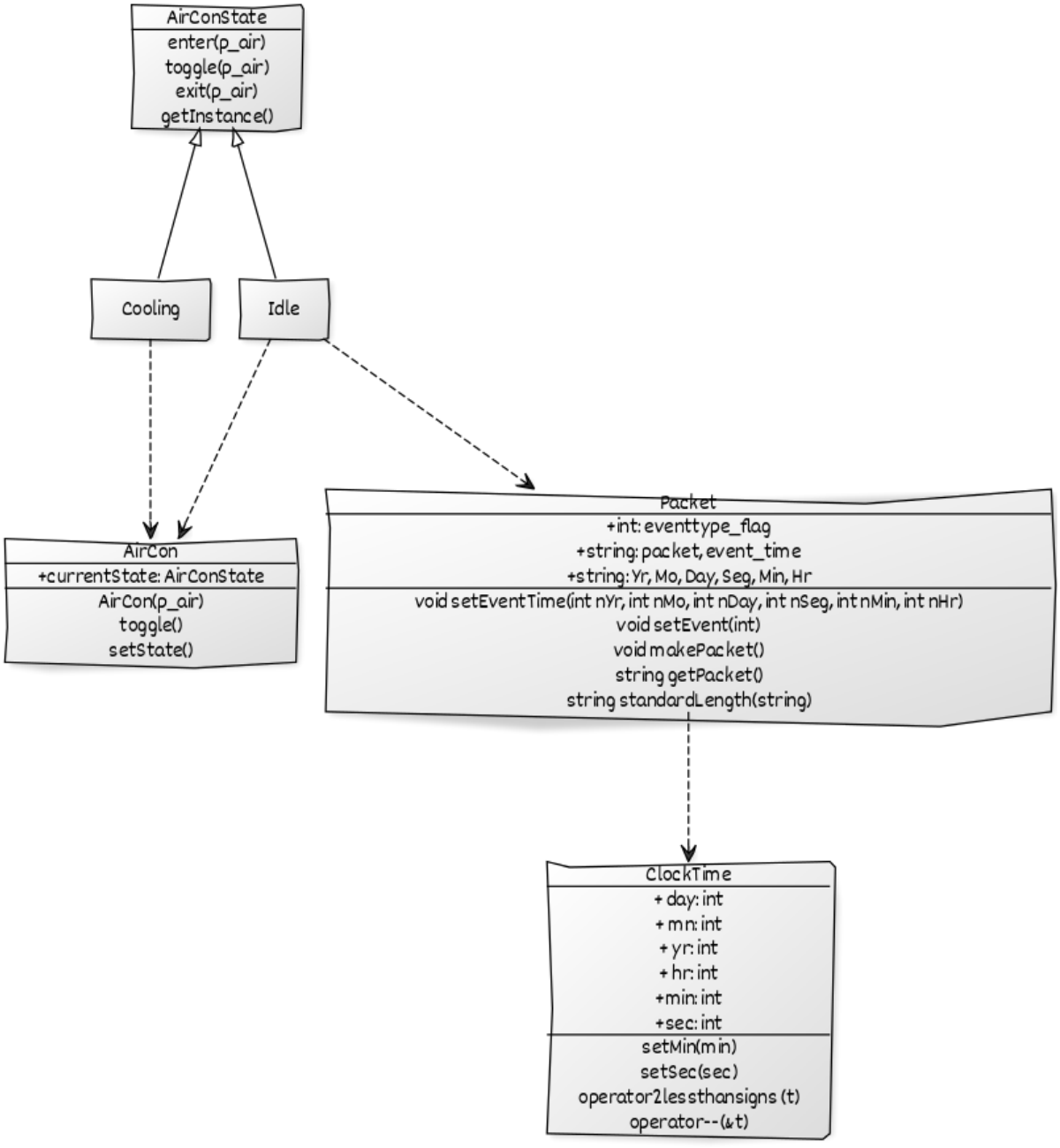
Aproveitamos dos dois cores do RP2040 para implementar a máquina de estados finitos (abaixo) num e a cronometragem necessária para fornecer o momento que um evento de log acontece no outro. Similarmente, os requerimentos de projeto exigem duas interrupções por hardware: do acionamento do botão para conectar o ar-condicionado e da ligação do UART para transferir os dados do sistema embarcado para um notebook. Por este motivo, implementamos uma interrupção em cada core. Embora a cronometragem seja interrompida para gravar um evento no log, esta interrupção é tão breve que não afeta a cronometragem de um segundo. Finalmente, o software do sistema embarcado é compilado no compilador GCC arm-none-eabi. O toolchain é um grupo de ferramentas específicas para Arm Cortex-A, Arm Cortex-M e Arm-Cortex-R de 32 bits e utiliza o compilador GCC.



Máquina de estados finitos para o sistema embarcado

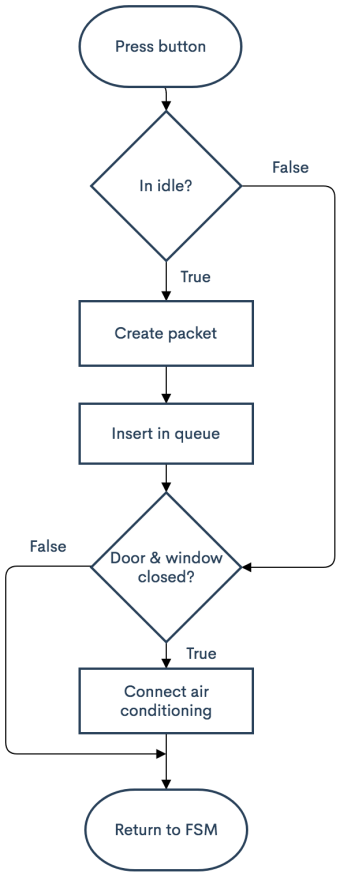
Software do sistema embarcado

Quanto ao software do sistema embarcado, implementamos o sistema com C++ e sua abordagem de orientação de objetos para organizar a máquina de estados finitos e o string que é empacotado e enviado por UART, se bem que utilizamos código procedural em algumas seções da máquina de estados finitos, como as interrupções devido ao hardware. O diagrama de classes tem a seguinte forma:

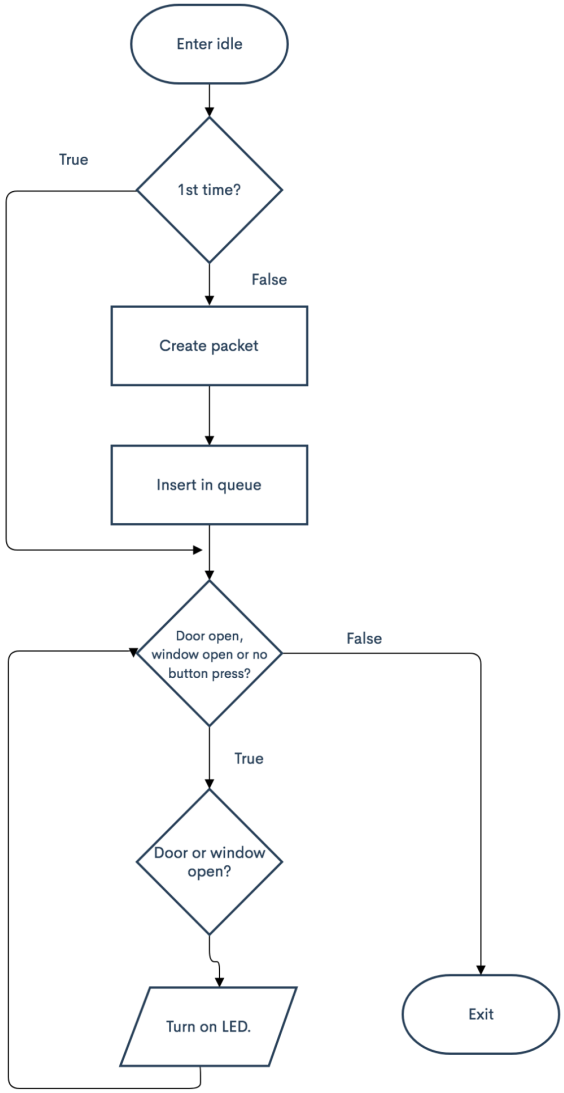

AirConState
enter(p_air)
toggle(p_air)
exit(p_air)
getInstance()
Cooling
Idle
AirCon
+currentState: AirConState
AirCon(p_air)
toggle()
setState()
Packet
+int: eventtype_flag
+string: packet, event_time
+string: Yr, Mo, Day, Seg, Min, Hr
void setEventTime(int nYr, int nMo, int nDay, int nSeg, int nMin, int nHr)
void setEvent(int)
void makePacket()
string getPacket()
string standardLength(string)
ClockTime
+ day: int
+ mn: int
+ yr: int
+ hr: int
+min: int
+sec: int
setMin(min)
setSec(sec)
operator2lessthansigns (t)
operator--(&t)
CREATED WITH YUML

Algumas seções procedurais cruciais foram representadas como fluxogramas:

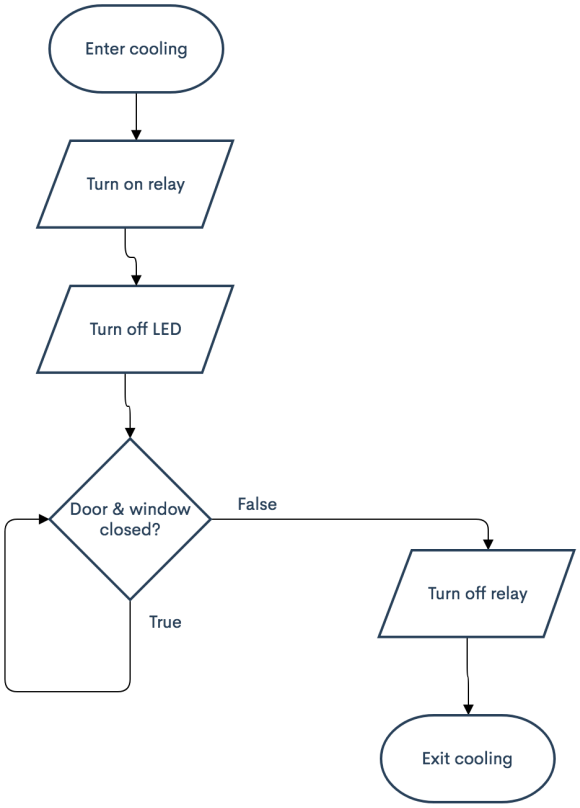
Button press



Estado Idle



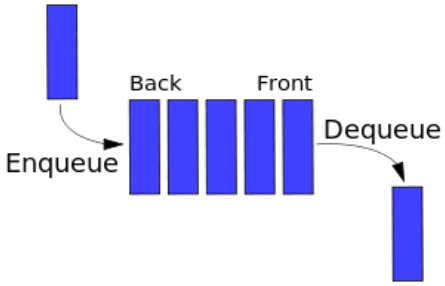
Estado Cooling



Devido à extração de dados pelo utilizador, é necessário descrever um programa que possa ser ativado num notebook para esta extração de dados. Este programa valida a identidade do utilizador e mostra um menu das opções especificadas na seção anterior. Logo que o administrador está validado e a conexão de UART está estabelecida, o administrador recebe a opção de baixar o log de eventos para o seu notebook. Se o administrador aceitar, os dados são enviados. Os dados que são necessários para realizar as três opções detalhadas nos requerimentos e os seus tipos, em parênteses, são:

1. Um código único para identificar o dispositivo (int);
2. O tipo de eventos, dos dois descritos nos requerimentos (int);
3. Se o evento resulta no começo do esfriamento (bool);
4. Se o evento resulta na terminação do esfriamento (bool);
5. A data e a hora do evento (5 ints).

A fila, organizada como *first in, first out* (fifo), precisa de dois ponteiros para indicar o início da fila para a adição de novas entradas e para indicar o fim da fila para a remoção da primeira entrada quando é chamada para sair. Assim, a fila tem o seguinte formato:



(fonte: https://en.wikipedia.org/wiki/Queue_(abstract_data_type)

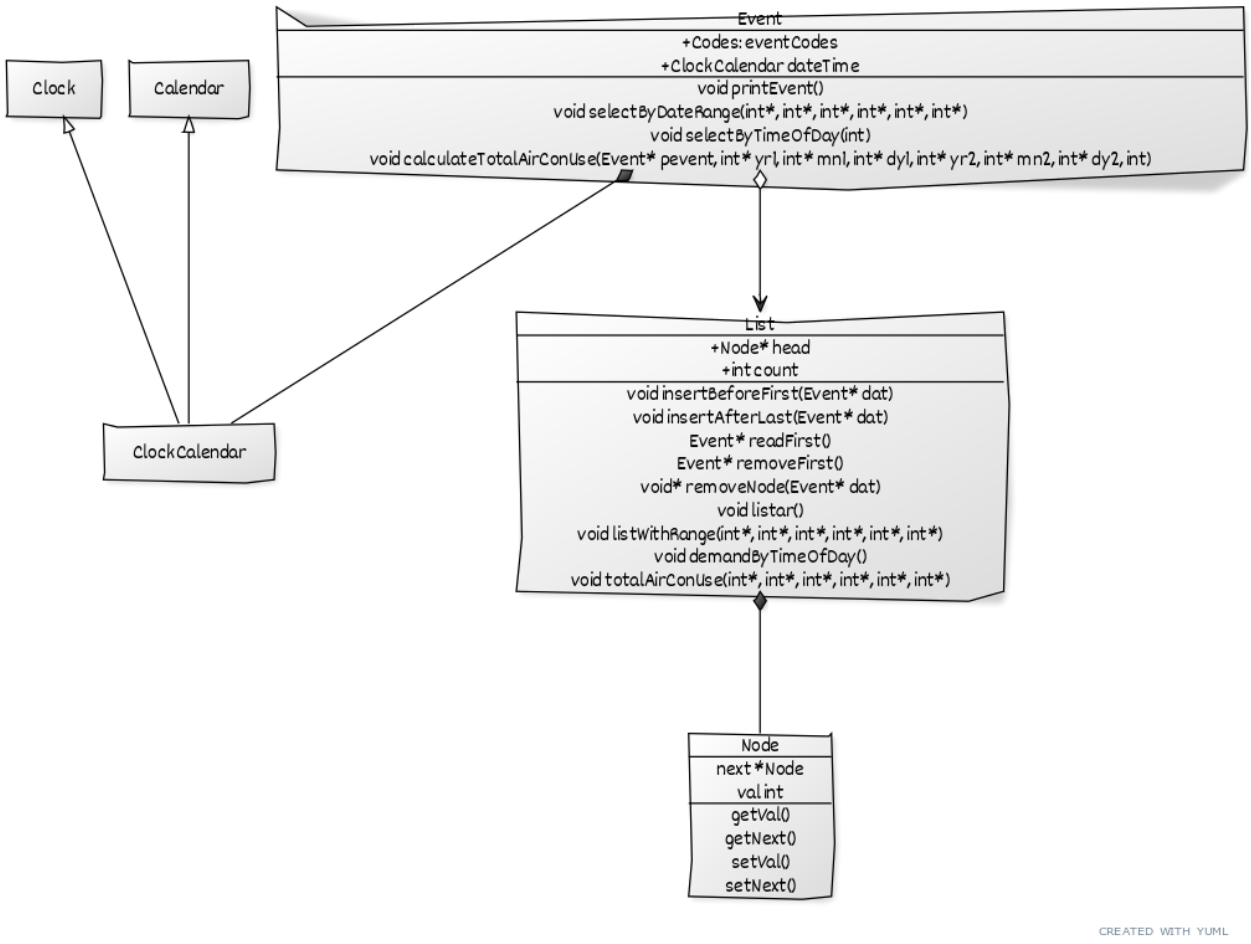
Da forma a facilitar o envio destes dados, todos são convertidos em char e depois colocados num string. Como todos os strings têm o mesmo comprimento, não há necessidade de especificar isto, apenas verificar que o string completo chega. Os strings são guardados numa fila no microcontrolador e numa lista encadeada no notebook, onde são manipulados para produzir as opções previamente especificadas.

Software do notebook

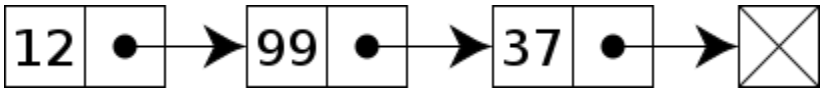
Novamente, implementamos o software do notebook em C++ com a sua abordagem de orientação de objetos, apesar de implementar a recepção de dados pelo UART proceduralmente. O software foi compilado com a versão de g++ da LLVM no MacOS e o IDE usado foi Xcode.

Em termos das bibliotecas usadas, é necessário, novamente, usar <iostream> e <string>. No entanto, várias outras bibliotecas foram usadas. Primeiro, <vetor> é usado para organizar os dados para que possam ser usados para calcular o tempo total de uso num intervalo - porém, isto poderia ser feito num array simples. Para a comunicação serial, a biblioteca <fcntl.h> é necessário para alguns *requests* da função *open();* <unistd.h>, para funções *write(), read(), close()* e o tipo *ssize_t*, que permite retornar -1 como um sinal de erro de uma chamada a uma função; <termios.h>, alterar parâmetros como baud; <chrono>, para usar funções que esperar até que todos os dados sejam transmitidos na comunicação UART .

O diagrama de classes é:



No programa do notebook, implementamos uma lista encadeada, que é uma lista que mostra a relação entre uma entrada na lista e os seus vizinhos. Assim, possibilita mais opções de manipulação dos dados, além de simplesmente acrescentar ou remover entradas. No entanto, estas possibilidades implicam uma complexidade maior. Tem o seguinte formato:



(fonte: https://en.wikipedia.org/wiki/Linked_list)

Software para celular

O software para o aplicativo móvel será desenvolvido para Android com C++ no MacOS. Como já temos Xcode instalado, vamos instalar Java JDK 6, Apache ANT Build System e GNU Make para implementar este aplicativo que terá funções parecidas com as do sistema de notebook.