Janela Automatizada

Karla Justen e Luan Lázaro Vieira

Motivação

Com as atividades do dia-a-dia ocupando a maior parte do tempo, é de interesse comum retornar a residência com ela confortável e agradável. Para conseguir isso, uma abordagem é manter o ambiente arejado, o que pode ser realizado com a automatização das janelas, permitindo abrir e fechar com base em condições como temperatura, umidade, concentração de CO2. A qualidade do ar proveniente de uma circulação natural mostra-se muito maior do que as de locais que são controlados com aquecedores e similares (8)(11), a boa qualidade do ar com boa circulação evita problemas respiratórios e alergias (3). Em relação ao aspecto humanitário, janelas que controlam-se sozinhas são excelentes para pessoas idosas e portadores de deficiência (3).

Objetivos

Desenvolver um sistema capaz de simular a abertura e fechamento de uma janela com abertura deslizante horizontal, com base em parâmetros fornecidos pelo usuário e por sensores. Inicialmente os sensores usados serão: o termômetro; detector de chuva, e de luz.

Além desse objetivo principal, será desenvolvido um aplicativo mobile para que o usuário possa configurar o sistema remotamente, possibilitando ligar e desligar o sistema automático para noite e dias; definir a temperatura mínima e máxima de conforto do cômodo, considerando que abrir a janela irá resfriar o comodo; entre outras funcionalidades.

Por fim, o projeto deve ter uma iniciativa sustentável em relação a isso. Em relação à sustentabilidade, o sistema deve ser projetado de forma que consuma pouca energia e seja energizado com uma fonte de energia limpa, no caso, a solar.

Figura 01: Janela de abertura deslizante horizontal.

Metodologia

O projeto será desenvolvido com base no processo cíclico PDCA (1), considerado efetivo para a busca do aperfeiçoamento por ser um método de melhoria contínua. Esse processo consiste em 4 fases Planejar-Fazer-Verificar-Ajustar (Plan-Do-Check-Act, em inglês). Na primeira fase, Planejar, é feita a identificação de objetivos e os métodos que serão usados para atingi-los. Na segunda fase, Fazer (D de Do, em inglês), é feita a capacitação dos envolvidos para que sejam capazes de realizar a implementação do que foi planejado. Nessa etapa também é realizada a implementação. A terceira fase é Verificar (C de Check, em inglês), onde os resultados são avaliados, comparando-os com os esperados, se houver uma diferença é considerado um problema a ser resolvido, necessitando realizar análises para identificar as causas. Com base na fase anterior, Checar (A de Act, em inglês) consiste em aplicar ações corretivas ou de melhorias para evitar que os problemas se repitam. Chegando ao fim do ciclo, como esse processo é de melhoria contínua, o ciclo é retomado.

Tarefas

Tarefa 1:

- Planejar(Plan) - Procurar por trabalhos antigos, que utilizam dos mesmos recursos que pretendemos usar. Procurar por trabalhos de como calcular a energia necessária pelo projeto; Planejar implementação do sistema e do circuito a ser construído.
- Fazer(Do) - Relatório com tudo explicado.

Tarefa 2:

- Planejar(Plan) - Estudar o uso adequado no Raspberry Pi para três sensores: Presença de chuva, claridade e temperatura, mais controle de leds. Verificar quais valores e intervalos enviados por cada sensor.
- Fazer(Do) - Configurar o Raspberry Pi (instalando SO e preparando apara executar o sistema). Desenvolver a primeira versão do sistema, consistindo no controle do led (aceso = janela aberta; apagado = janela fechada) através de valores simulados dos sensores e por controle manual (botão).
- Verificar(Check) - Testar o comportamento da aplicação, simulando mudanças de temperatura, claridade e de presença de chuva e verificando se o dispositivo acompanha o comportamento esperado. Elaborando um relatório do teste incluindo os momentos que a simulação dos sensores se modificam e a respectiva resposta do sistema.
- Checar (Act) - Caso o relatório do teste seja insatisfatório, necessário agir para corrigir os problema encontrados.

Tarefa 3:

- Planejar (Plan) - Verificar se o planejamento dos circuitos dos sensores com o Raspberry Pi, feito na primeira tarefa, continua consistente com o estado atual do projeto
- Fazer (Do) - Montar o circuito e instalar os sensores físicos; Desenvolver código de leitura dos valores recebidos pelos sensores.
- Verificar (Check) - Criar condições para realizar mudanças nas leituras dos sensores, para conferir a reação do sistema. Criar um relatório baseado nas mudanças de temperatura, presença de chuva, claridade e no comportamento do dispositivo.
- Checar (Act) - Corrigir possíveis incoerências baseado no relatório de verificação.

Tarefa 4:

- Planejar(Plan) - Estudar técnicas para medir o consumo de energia da aplicação após a configuração do Raspberry Pi; Pesquisar placas solares no mercado, seus respectivos preços e quanto de energia é conseguido para sustentar o sistema.
- Fazer(Do) - Calcular efetivamente o consumo energético da aplicação (O cálculo deve contar com o uso de energia dos três sensores e do sistema em si). Fazer um relatório do tamanho, em m^2, necessário para a placa solar alimentar o sistema. Levando em consideração, também, o tempo necessário de exposição ao sol.
- Verificar (Check) - Verificar viabilidade de alimentar o projeto com energia solar.
- Checar (Act) - Caso seja inviável alimentar o projeto com apenas energia solar, analisar maneiras de tornar isso viável e se possível aplicá-las.

Tarefa 5:

- Planejar (Plan) - Estudar ambientes para desenvolver um aplicativo mobile para configurar remotamente o dispositivo, tal aplicativo deve ser capaz de desativar/ativar o fechamento/abertura automático da janela durante o dia e a noite; e configurar o intervalo de temperatura de conforto do comodo. Além de definir a forma de comunicação entre o aplicativo e Raspberry Pi.
- Fazer(Do) - Implementar o aplicativo.
- Verificar (Check) - Verificar se as alterações de configurações, via aplicativo, realmente causam mudanças no sistema.
- Checar (Act) - Caso não funcionar como esperado, realizar análise para verificar os causadores dos problemas e procurar corrigi-los.

Entregáveis

- Tarefa 1 - 02/10 - Planejamento completo.
- Tarefa 2 - 16/10 - Código C++ com a sinalização de abertura / fechamento da janela, utilizando sensores simulados. Relatório de teste da simulação de abertura/ fechamento;
- Tarefa 3 - 30/10 - Código C++ com a sinalização de abertura / fechamento da janela, utilizando sensores reais. Relatório apresentando o circuito montado com os sensores e os testes realizados e seus resultados;
- Tarefa 4 - 13/11 - Código do aplicativo para a configuração remota do sistema.
- Tarefa 5 - 04/12 - Relatório com informações do modo que foi calculado o custo de energia, o orçamento das placas solares com seus respectivos fabricantes, e com a área necessária de exposição à luz. O relatório deve concluir a viabilidade do uso de placas solares.

A seguir serão apresentada descrições do que foi feito para cada entregavel:

Entregavel 01:

Foi realizado planejamento completo, inicialmente foi pensado em usar EPOS Motes, gerando a primeira versão do planejamento, entregue na data prevista 02/10. Disponível em: https://drive.google.com/file/d/0B1nNj03qFNnYUjJvcEhObmF4eGM/view?usp=sharing

Mas devido a inexperiência do grupo em trabalhar com essa placa, foram realizadas alterações no planejamento, sendo utilizado Raspberry pi. Disponível em: https://drive.google.com/file/d/0B1nNj03qFNnYdS1rU2V5QWFWQ0U/view?usp=sharing

Entregavel 02:

Foi desenvolvida a implementação da lógica da classe Brain e utilizada como entrada os valores dos sensores de forma simulada. Para isso foram estudados os valores que realmente serão recebidos pelos sensores, para que a essa etapa seja preparada para a próxima etapa, quando forem adicionados os sensores.

Mais informações sobre o desenvolvido está em:
https://epos.lisha.ufsc.br/Janela+Automatizada#Software_de_Simula_o_e_seus_testes

O estado da wiki para este entregável:
https://drive.google.com/file/d/0B1nNj03qFNnYSTBFaExrcV9MaFU/view?usp=sharing

Entregavel 03:

A implementação da comunicação com os sensores foi implementada e aprimorada. O acesso aos seus dados é de forma idêntica ao realizado no projeto "automaticWindow_Simulation". Disponível em:
https://github.com/karlajusten/automaticWindow

Além disso, os relatórios gerados pelos testes em:
https://github.com/karlajusten/automaticWindow/tree/master/Testes
Como mostra o relatório, a leitura da temperatura ocorreu algumas vezes de forma errônea, tal erro foi corrigido ainda nesta etapa do projeto.

Um video foi feito apresentado um teste sendo realizado: https://youtu.be/hPiZOoP2CRU

Entregavel 04:

Nesta etapa foi utilizada a ferramenta App Inventor (http://appinventor.mit.edu/explore/) para desenvolver o aplicativo Android que permite alterar configurações da Janela Automatizada. A transferência de informações é feita através do servidor TinyWebDB (http://appinvtinywebdb.appspot.com/). Para acessar esse servidor, no raspberry pi foi utilizada a bibliteca curl.

Montamos uma versão de código para ser aplicado num contexto real, evitando comportamento estérico em https://github.com/Luanes/automaticWindowThread (não está completo).

Foi tabelado um orçamento para haver noção dos custo do projeto, na sessão "Orçamento"

Entregavel 05:

Esta etapa foi substituida pelo Entregavel Extra, apresentado a seguir.

Entregavel Extra:

Foi construído um protótipo de janela, apresentado em "Projeto de construção do Protótipo de Janela". Além dos dispositivos já apresentados, foi acrescentado um servo motor de rotação contínua (originalmente era de 360º fora realizadas modificações no motor, como apresentado em "Servo Motor de Rotação Contínua") e dois interruptores, para sinalizar quando estiver aberto ou fechada a janela. Para alimentar o servo motor é necessário uma bateria de 7V mais um regulador de tensão (para reduzir para 5V).

Importante: Como o servo motor é controlado por pino PWM, é preciso executar o projeto em sudo.

Planejamento

Task	25/09	02/10	09/10	16/10	23/10	30/10	06/11	13/11	27/11	04/12
Tarefa1	x	D1								
Tarefa2		x	x	D2						
Tarefa3			x	x	x	D3				
Tarefa4						x	x	D4		
Tarefa5								x	x	D5

Protótipo do Projeto (Hardware)

Neste projeto serão utilizados três sensores: um resistor dependente de luz para detectar claridade; um sensor de temperatura; e um sensor de chuva YL83. Esses sensores irão auxiliar o sistema na tomada de decisão de quando abrir ou fechar a janela.

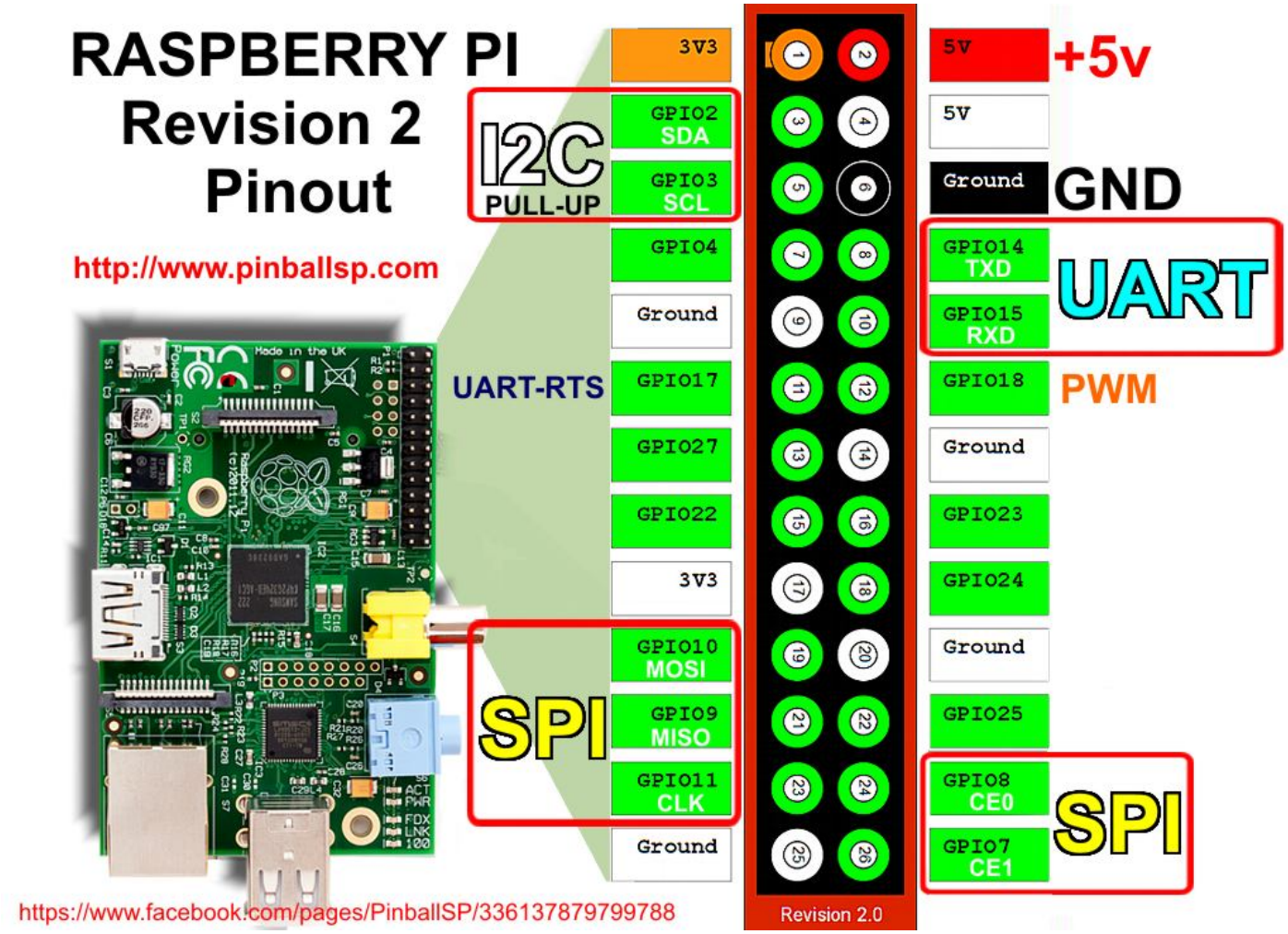
Para sinalizar as ações que serão enviadas para a janela, será utilizado um led, que quando aceso representa a janela aberta e apagado como fechado. Já para representar a utilização manual da janela, um botão é usado, apagando ou acendendo o led.

Para acessar os dados dos pino GPIO, há diferentes bibliotecas:

1. http://wiringpi.com/
2. https://github.com/JoachimSchurig/CppGPIO/;
3. http://hertaville.com/introduction-to-accessing-the-raspberry-pis-gpio-in-c.html

A biblioteca escolhida para ser usada foi a WiringPi.

Mapa dos pinos GPIO do Raspberry pi 1 B utilizado:



A seguir é apresentado as configurações feitas com o Raspberry Pi. Após isso são apresentados os sensores utilizados neste projeto e informações sobre eles

Instalação do SO no Raspberry Pi

Foi necessário realizar a instalação do Sistema Operacional Raspbian no cartão de memória do Raspberry Pi. Para isso foram realizados os seguintes passos:

1º Baixei NOOBS: Offline and network install; Version:2.4.4; Release date:2017-09-08 do link https://www.raspberrypi.org/downloads/noobs/
2º Descompactei SO; coloquei os arquivos no cartão SD; despluguei o cartão SD do pc;
3º COnectei o monitor HDMI, o mouse USB, o teclado USB e o cartão de memória no Raspberry Pi;
4º Fiz o processo apresentado em “Instalação do Sistema Operacional” do site: https://www.filipeflop.com/blog/tutorial-raspberry-pi-linux/
(iniciou a instalação, demorou em torno de 40 minutos)
5º Cliquei ok na mensagem de finalização. Reiniciou o SO e apareceu a área de trabalho.

Cuidado Importante: Lembre-se que a USB do computador suporta no máximo 500 mA, portanto não ligue o Raspberry diretamente nessa porta, pois o Rpi precisa de corrente de 2A.

Conectando Raspberry Pi à Internet via cabeamento de rede com o Notebook

Foram seguido os passos apresentados em (24):

1º Instalar o network-manager: sudo apt-get install network-manager;

2º Instalar o nmap: sudo apt-get install nmap;
3º Alterei a configuração IPV4 da rede cabeada para: “compartilhada com outros computadores”: System Sttings → Network → Wired → Options → IPv4 Settings → Method: “Share to other computers”; reiniciei o pc;
4º Conectei o cabo de rede entre Raspberry pi e notebook
5º Verifiquei o endereço de transmissão da conexão ethernet:
$ /sbin/ifconfig eth0 | grep "Bcast" | awk -F: '{print $3}' | awk '{print $1}';
Retornou o endereço ip: 10.42.0.255
6º Utilizar essa informação para descobrir o IP do raspberry pi:
$ nmap -n -sP 10.42.0.255/24;
O IP do raspberry pi é 10.42.0.71

Agora o Raspberry pi tem acesso a internet, via notebook.

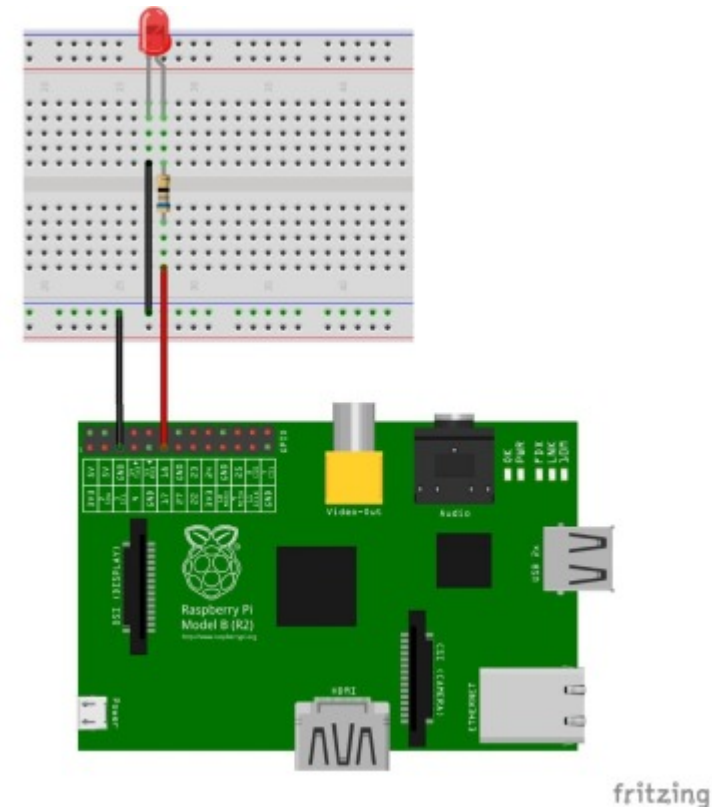
Instalando a Bilbioteca WiringPi no Raspberry pi

Para montar um código em c++ com auxílio da biblioteca wiringPi. É preciso instalá-la, para isso foram seguidas as instruções apresentadas neste video (https://www.youtube.com/watch?v=J6KsTz6hjfU).
Gerando os seguintes passos:
1º precisei baixar o git: $ sudo apt-get install git
2º baixar a biblioteca wiringPi:
$ git clone git://http://git.drogon.net/wiringpi
(pode ser pq a net esta lenta, essa parte demorou bastante... pq nao marquei o tempo? caramba esta demorando...)
3º Entrar na pasta wiringPi e executar o arquivo build, com os seguintes comandos:
$ cd wiringPi
$ ./build
4º Montei o circuito apresentado no vídeo, com o positivo energizado pelo pino GPIO 17.
5º Entrei na pasta examples e abri o arquivo blink.c no terminal com o comando:
$ cd examples
$ nano blink.c
6º fiz as alterações apresentadas no vídeo (não é necessário);
7º compila o arquivo blink.c e executa :
$ gcc blink.c -o blink -l wiringPi
$ sudo ./blink
Se foram feitas as alterações sugeridas no vídeo, o led irá piscar 5 vezes e parar. Se não fizer, o led piscara indefinidamente, parando apenas quando interromper a execução

LED

O LED será utilizado para representar o estado da janela: aberta/ON; fechada/OFF. Foi o primeiro experimento realizado com o Raspberry Pi, pois é considerado o "Hello Word" ao lidar com GPIO do Raspberry Pi.

Após montar o seguinte circuito, é utilizado um resistor de 27 ohms:



Foi compilado e executado o arquivo blink.c salvo em: https://github.com/karlajusten/automaticWindow/tree/master/Examples, que acende e apaga o led por 5 vezes. Esse arquivo é uma leve alteração do código blink.c dos exemplos oferecidos pela biblioteca WiringPi.

Com base na biblioteca wiringpi, salvo no arquivo Sensor.cpp, ficando da seguinte maneira:

```
void Sensor::openWindow(){
  digitalWrite (LED, HIGH);
  delay (500);
}

void Sensor::closeWindow(){
  digitalWrite (LED, LOW);
  delay (500);
}

bool Sensor::isWindowOpen(){
  if(digitalRead(LED))
        return true;
  return false;
}
```

Onde nos métodos openWindow() e closeWindow() é alterado o estado do LED, acendendo (HIGH) e apagando (LOW), respectivamente. Já para verificar o estado do LED, há o método isWindowOpen().

Sensor de Temperatura (DHT11)

O sensor usado possui 4 pinos, com alimentação de 5V, que responde com 40bits, onde inclui a temperatura e a humidade medida. (19) Como não é útil a humidade do ambiente, essa informação será ignorada.

No trabalho “Sistema de Monitoramento e Controle Adaptável ao Contexto-Aware” (2)foi utilizado DHT11:

“The measurements of external temperature and humidity were made using the DHT11 sensor. Different from the LDR, that is analog, the DHT11 is a digital transducer. The output pin of the DHT11 is connected to a digital GPIO pin of the EPOSMoteIII. The MCU must send a start signal to ask the transducer for the information. The transducer answers with a signal that diferentiates 0 from 1 according to the duration of high.
When EPOSMoteIII sends a start signal, DHT11 changes from the low-power-consumption mode to the running-mode, waiting for the MCU to complete the start signal. Once it is completed, DHT11 sends a response signal of 40-bit data that include relative humidity and temperature information to the MCU. Without the start signal from MCU, DHT11 will not give back the response signal. Once data is collected, DHT11 will change to low power-consumption mode until it receives a start signal from MCU again. The overall communication process between the transducer and the micro-controller is shown on the figure below.”

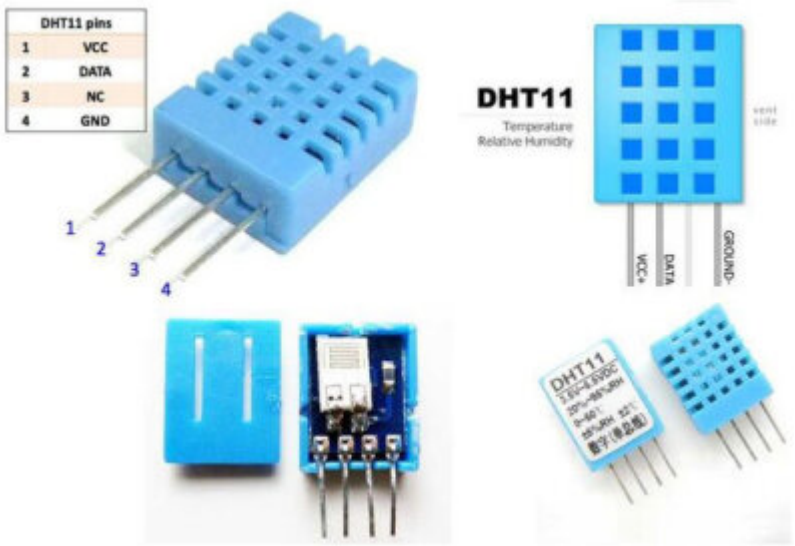
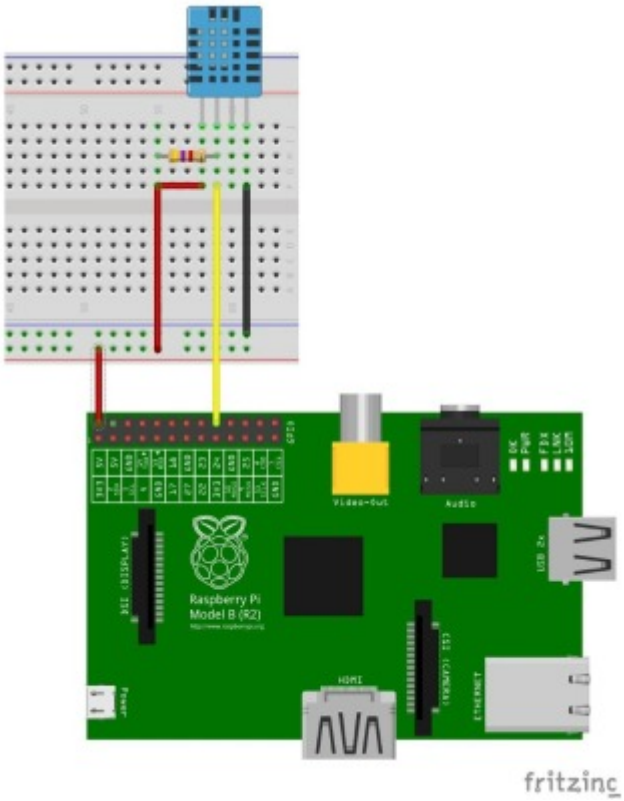


Figura 04: DHT11 de 4 pinos

Para realizar um experimento inicial foi montado o circuito a seguir, com um resistor de 10Kohms:



Para conseguir programar é preciso saber como esse sensor funciona. Segundo (22), o dado transmitido tem o seguinte formato:

8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum

onde RH = Indice de Humidade; T = temperatura.

O que é de interesse são os dois conjuntos de 8 bits de temperatura, sendo que o primeiro (da esquerda para direita) é o valor inteiro da temperatura e o segundo os decimais.

Segundo (22), o seguinte procedimento precisa ser realizado apra inicializar o DHT11:
“When the communication between MCU and DHT11 starts, MCU will pull down the DATA pin for least 18ms. This is called “Start Signal” and it is to ensure DHT11 has detected the signal from MCU. Then MCU will pull up DATA pin for 20-40us to wait for DHT11’s response”

De saída são gerados valores entre 0 e 50 graus Celsius.

Assim foi testado o código apresentado por (22), salvo como dht11Sensor.c com algumas alterações. Apresentado em https://github.com/karlajusten/automaticWindow/tree/master/Examples.

Para acrescentar o código apresentado por (22) no projeto, ele foi adaptado. Apesar de não alterar a lógica, alguns erros de leitura da temperatura ocorriam. Não tendo resultado satisfatório. Realizamos uma pesquisa, foi realizado teste com outro sensor (com a suspeita do problema ser com o dispositivo), mas nenhum causador do erro pareceu. Então remontamos o código, só que deixando com a estrutura mais parecida com a do exemplo (sem mudar a lógica), isso foi o suficiente para resolver o problema de leitura.

```
int Sensor::getTemperature(){
        int count = 0;
        int temp = -1;
        while ( count < 50)
        {
                temp = aux_getTemperature();
                if (temp >=0 && temp <=50){
                        return temp;
                }
                delay( 1000 );
                count++;
        }

        return -1;
}
```

```
int Sensor::aux_getTemperature(){
        uint8_t laststate       = HIGH;
        uint8_t counter         = 0;
        uint8_t j               = 0, i;
        float   f;

        dht11_dat[0] = dht11_dat[1] = dht11_dat[2] = dht11_dat[3] = dht11_dat[4] = 0;

        pinMode( DHTPIN, OUTPUT );
        digitalWrite( DHTPIN, LOW );
        delay( 18 );
        digitalWrite( DHTPIN, HIGH );
        delayMicroseconds( 40 );
        pinMode( DHTPIN, INPUT );

        for ( i = 0; i < MAXTIMINGS; i++ )
        {
                counter = 0;
                while ( digitalRead( DHTPIN ) == laststate )
                {
                        counter++;
                        delayMicroseconds( 1 );
                        if ( counter == 255 )
                        {
                                break;
                        }
                }
                laststate = digitalRead( DHTPIN );

                if ( counter == 255 )
                        break;
                if ( (i >= 4) && (i % 2 == 0) )
                {
                        dht11_dat[j / 8] <<= 1;
                        if ( counter > 16 )
                                dht11_dat[j / 8] |= 1;
                        j++;
                }
        }

        if ( (j >= 40) &&
             (dht11_dat[4] == ( (dht11_dat[0] + dht11_dat[1] + dht11_dat[2] + dht11_dat[3])) ) )
        {
                return dht11_dat[2];
        }else  {
                return -1;
        }
}
```

Sensor de Chuva - YL83

O conjunto utilizado é formado “por uma placa que forma o sensor propriamente dito, com várias trilhas nos dois lados e material resistente à oxidação, que se encarrega de detectar o nível de chuva/umidade do ambiente. Esta placa, por sua vez, é ligada por meio de 2 fios ao módulo principal, que contém o circuito de controle que vai se comunicar com o microcontrolador.” (15)

“Quando o clima está seco a saída do sensor fica em estado alto e quando há uma gota de chuva, a saída fica em estado baixo. O limite entre tempo seco e chuva pode ser ajustado através do potenciômetro presente no sensor.” (16) Ou seja, quando retorna 0 significa que está chovendo; qualquer valor diferente, o clima está seco.

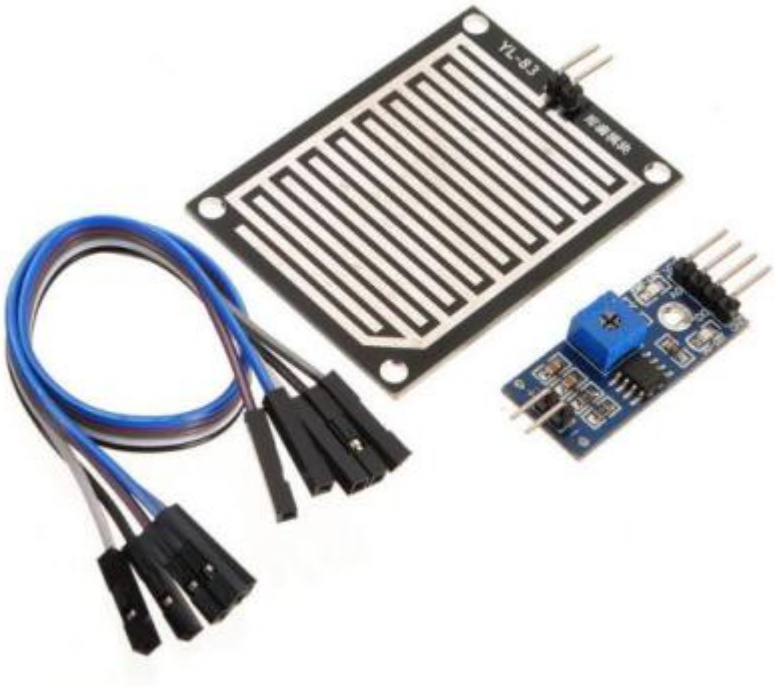


Figura 05: Sensor de Chuva YL83 utilizado (16)

Como experimento inicial foi rodado o exemplo apresentado em (23), os resultados foram muito bem sucedidos. Esse código está salvo como rainSensor.c em https://github.com/karlajusten/automaticWindow/blob/master/Examples.

Para receber as leituras dese dispositivo é tão simples quanto a leitura de LED', como implementado na classe Sensor.cpp:

```
bool Sensor::isRaining(){
  if(digitalRead(RAIN))
        return false;
  return true;
}
```

Sensor de Claridade - Resistor dependente da luz (LDR)

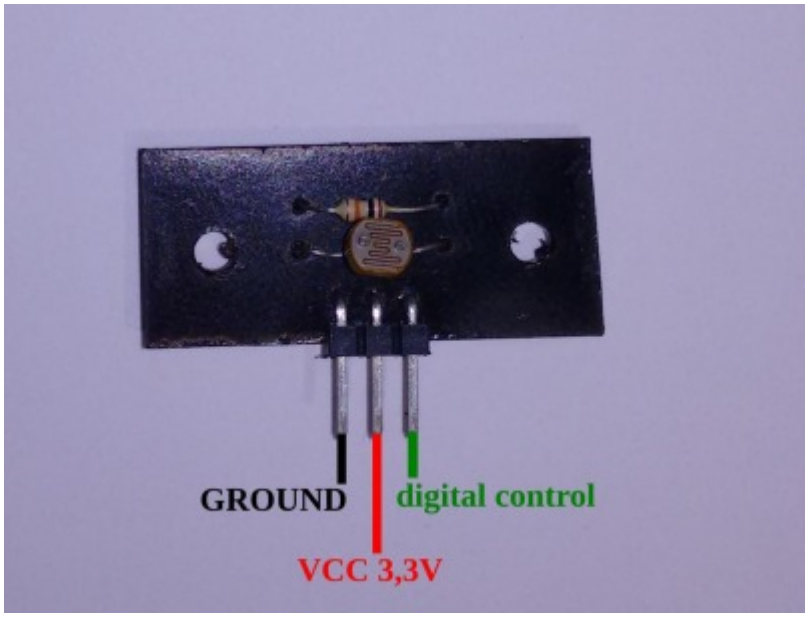
O sensor usado é um Resistor dependente da luz (LDR) de 5mm, onde sua resistecia aumenta quanto mais escuro está; consequentemente, a resistência reduz, quanto mais claro estiver. (18)

Figura 03: Resistor dependente da luz (LDR) de 5mm

No trabalho “Sistema de Monitoramento e Controle Adaptável ao Contexto-Aware” (2) foi utilizado LDR: “To measure luminosity we chose to use a Light-Dependent Resistor (LDR) of 5mm. Both the LDR and the

circuit schematic ,which was used for the measurements connected to an analog input of EPOSMoteIII, are illustrated in the Figure Below. The EPOSMoteIII has a 12-bit ADC converter for the analog inputs. The value read from the analog input in the mote, to which the LDR is connected to is sent to the gateway and mapped to values ranging from 0% to 100%.” (2)

O LDR disponibilizado pelo LISHA é o apresentado a seguir:



Para leitura dos seus dados, é utlizado o método "digitalRead(LDR)" do wiringPi, onde LDR foi definido como 25, o pino de "digital control" do sensor está conectado. Ficando da seguinte maneira:

```
bool Sensor::isDay(){
  if(digitalRead(LDR))
        return true;
  return false;
}
```

Onde "digitalRead(LDR)" retorna verdade se estiver em ambiente claro; false, se não.

Servo Motor de Rotação Contínua

O servo motor que a disposição é o TowerPro SG 5010, que não é de rotação contínua. Para servir ao propósito, foram feitas alterações para que ficasse de rotação contínua. Para isso, foram seguidas as instruções dadas por:
https://www.filipeflop.com/blog/alterando-servo-towerpro-sg-5010-para-rotacao-infinita/

Para controlar o motor, foi seguido os comandos dado por https://learn.adafruit.com/adafruits-raspberry-pi-lesson-8-using-a-servo-motor/software .

É possivel controlar via terminal, com os seguintes comandos:
gpio -g mode 18 pwm
gpio pwm-ms
gpio pwmc 192
gpio pwmr 2000
gpio -g pwm 18 100
Onde o ultimo comando controla a rotação e sua velocidade.

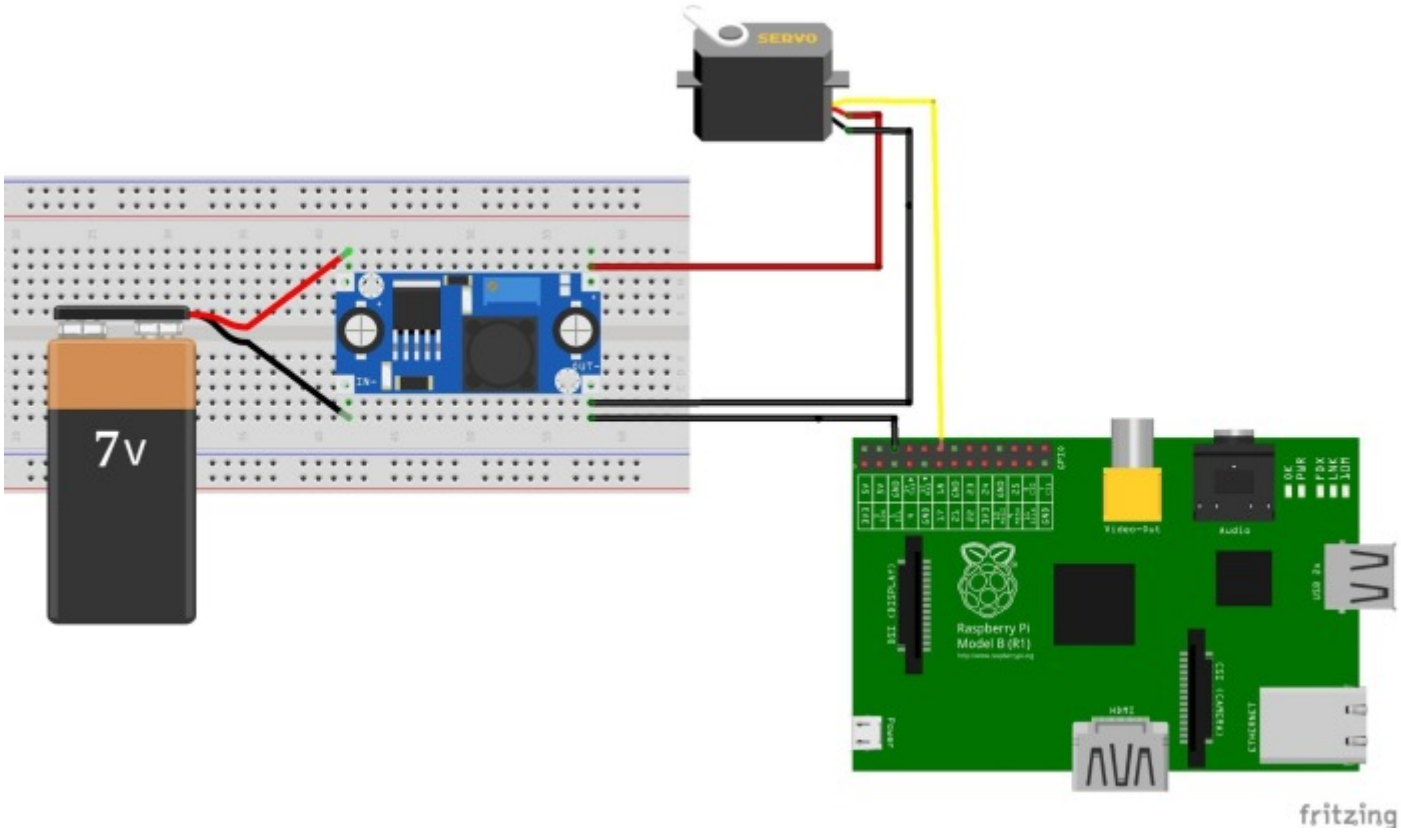
Foram testados os comandos de controle de 0 a 180, chegando a seguinte conclusão:

- 0: fica parado
- 1 - 150: sentido horário
- 156-157: fica para
- 158 - 180: sentido anti-horário

Agora, aplicando o controle do motor no contexto da janela, apresentado em https://github.com/karlajusten/automaticWindow/blob/master/Sensor.cpp

Importante: Para executar um código com GPIO no modo PWM é preciso realizá-lo em sudo. Se não, o raspberry pi trava e obriga a reinicialização (demoramos para perceber isso ¬¬).

O circuito de alimentação do motor precisou ser externalizada, pois apesar do raspberry pi oferecer GPIO com 5V, o motor precisa de mais corrente para rotacionar o motor. Assim é utilizada uma bateria de 7V e um regulador de tensão para reduzir a tensão para 5V. Ficando o circuito a seguir:

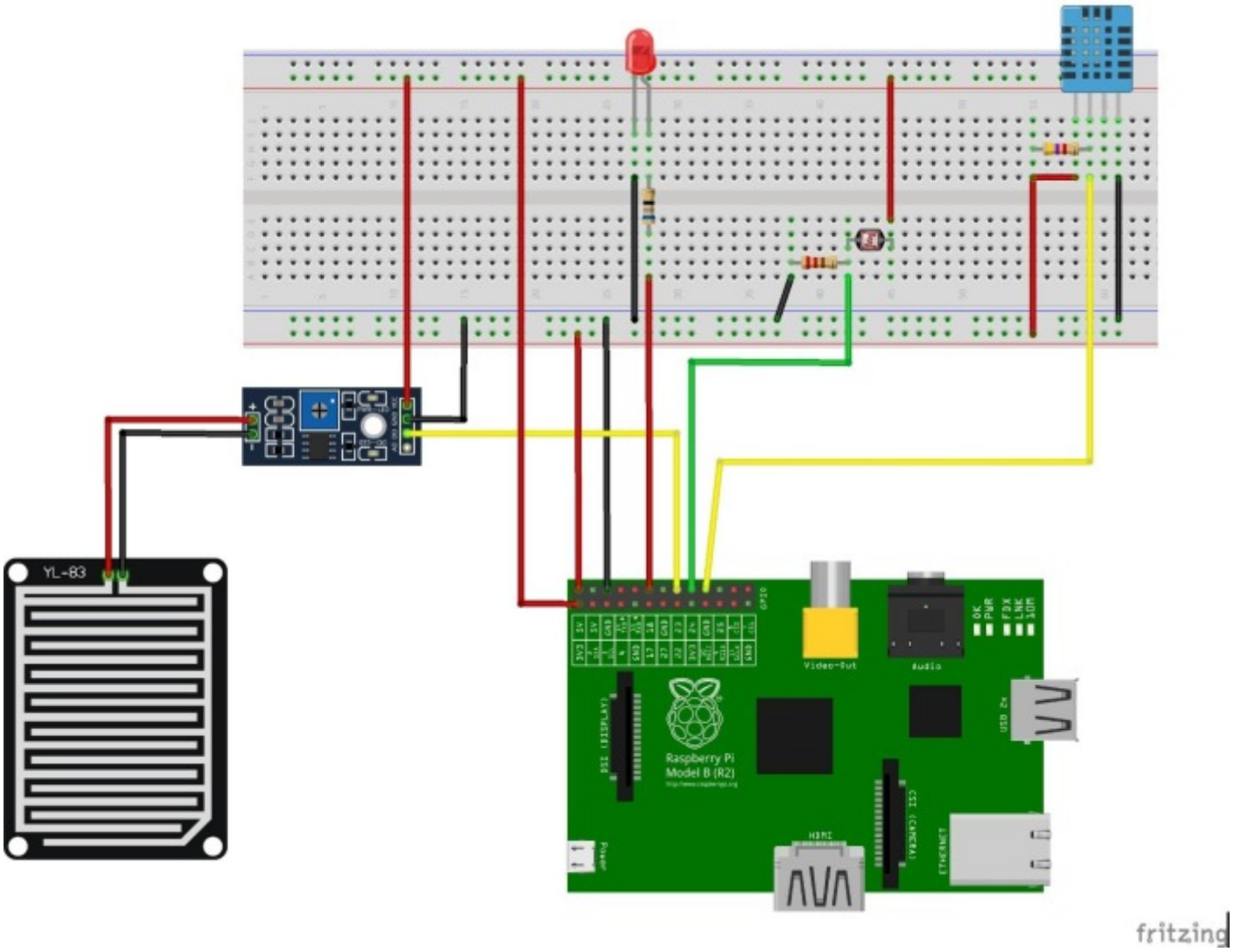


Botão (Switch)

No protótipo da janela para sinalizar quando a janela abriu ou fechou completamente são utilizados dois *switchs*, um em cada extremidade. Além de ser útil para o software, é importante para desligar/parar a rotação do motor, evitando desperdício de energia e queimar o motor.

Protótipo com todos os Sensores

Juntando os sensores, conseguimos o seguinte circuito:



O código desenvolvido que realiza comunicação entre Raspberry Pi e Sensores está em: https://github.com/karlajusten/automaticWindow

Software

Toda a tomada de decisão deve vir através do software. Como mencionado anteriormente, temos como motivação o aumento de qualidade de vida para quem utilizar a aplicação. Para isso, a aplicação precisa tomar decisões a partir das preferências do usuário. De maneira geral, o usuário deve optar por operar a janela de modo manual ou automático. Se for automático, as seguintes perguntas devem ser respondidas.

Quando a janela deve fechar?

- Na ocorrência de chuva;
- Temperatura do ambiente for MENOR do que a temperatura mínima que o usuário deseja aceitar.

Quando a janela deve abrir?

- Temperatura do ambiente for MAIOR do que a temperatura máxima configurada.

Modo automático será separado em duas partes:

- Automático de dia;
- Automático de noite;

O dispositivo é considerado em modo “automático” quando o modo automático estiver ligado ao mesmo tempo que seu momento regente no dia (dia e noite), suas transições devem ser como mostra a Figura 06.

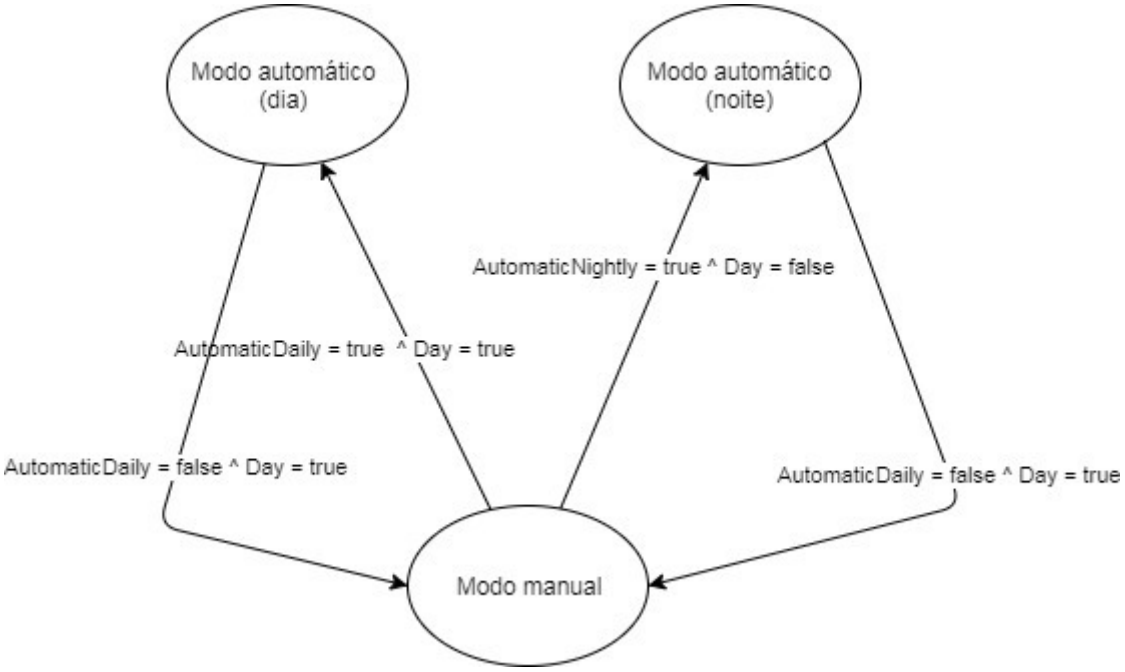


Figura 06: Diagrama de estados do modo automático

Levando em consideração as transições em relação aos estados “Aberto” e “Fechado” em junção com a necessidade do sistema estar em modo automático, obtemos o diagrama da Figura 07.

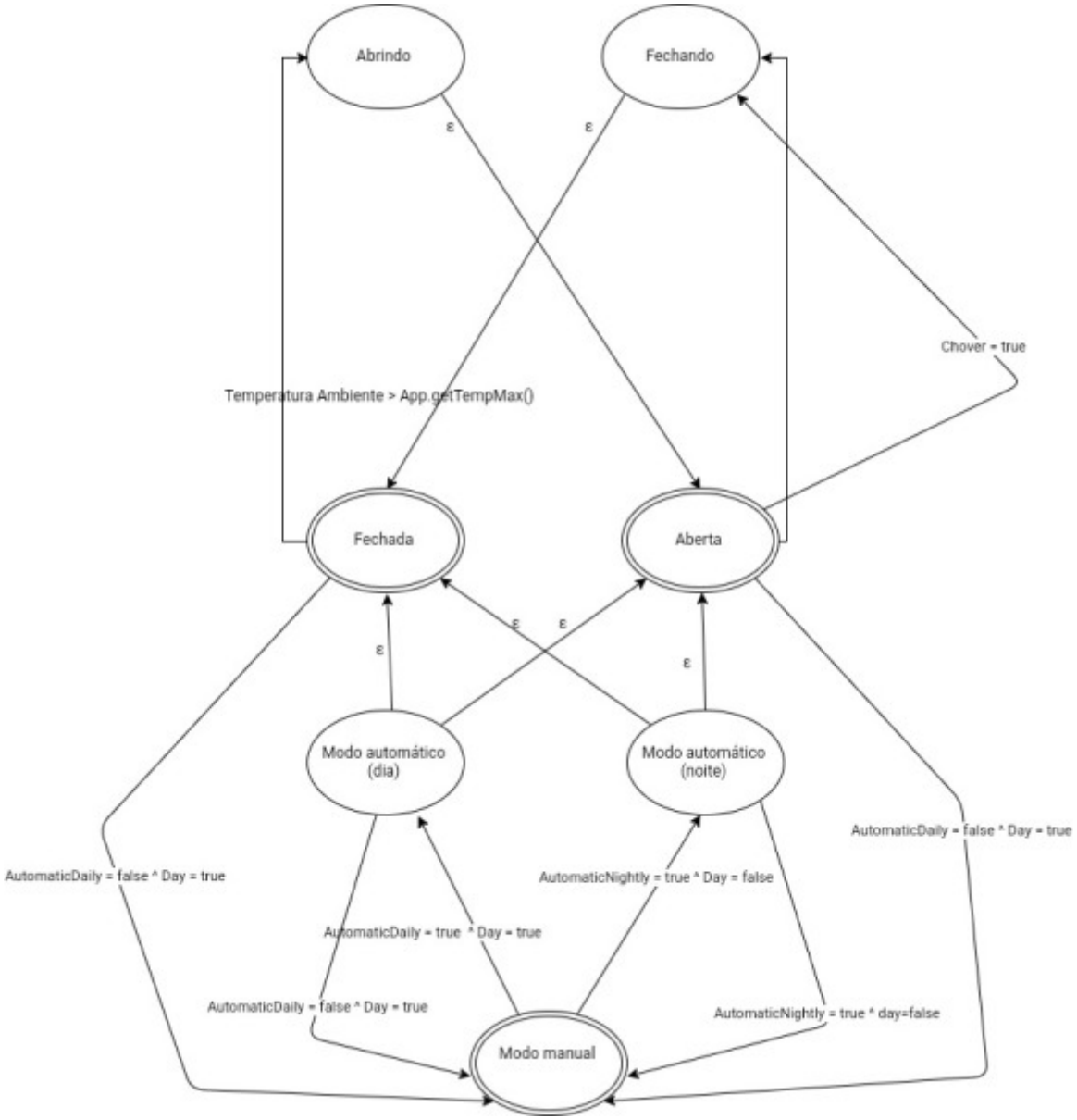


Figura 07: Diagrama completo

Para o controle do dispositivo, iremos utilizar uma classe chamada “Brain”, na qual será configurada para decidir as ações do dispositivo. A ordem de condições a serem satisfeitas antes de uma ação devem seguir o fluxograma da Figura 08.

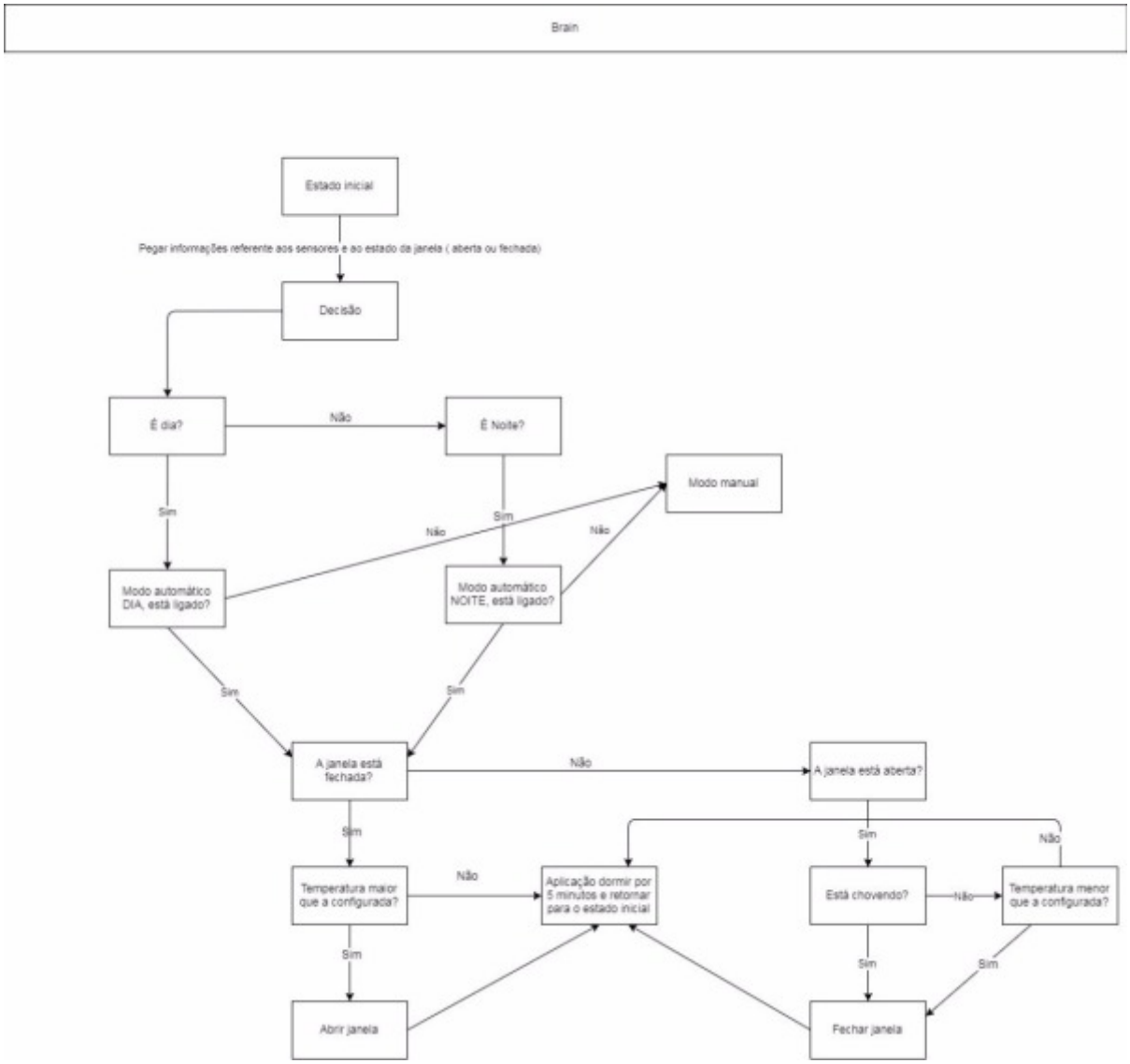


Figura 08: Fluxograma para o Brain.

A aplicação deve seguir o seguinte diagrama de classes:

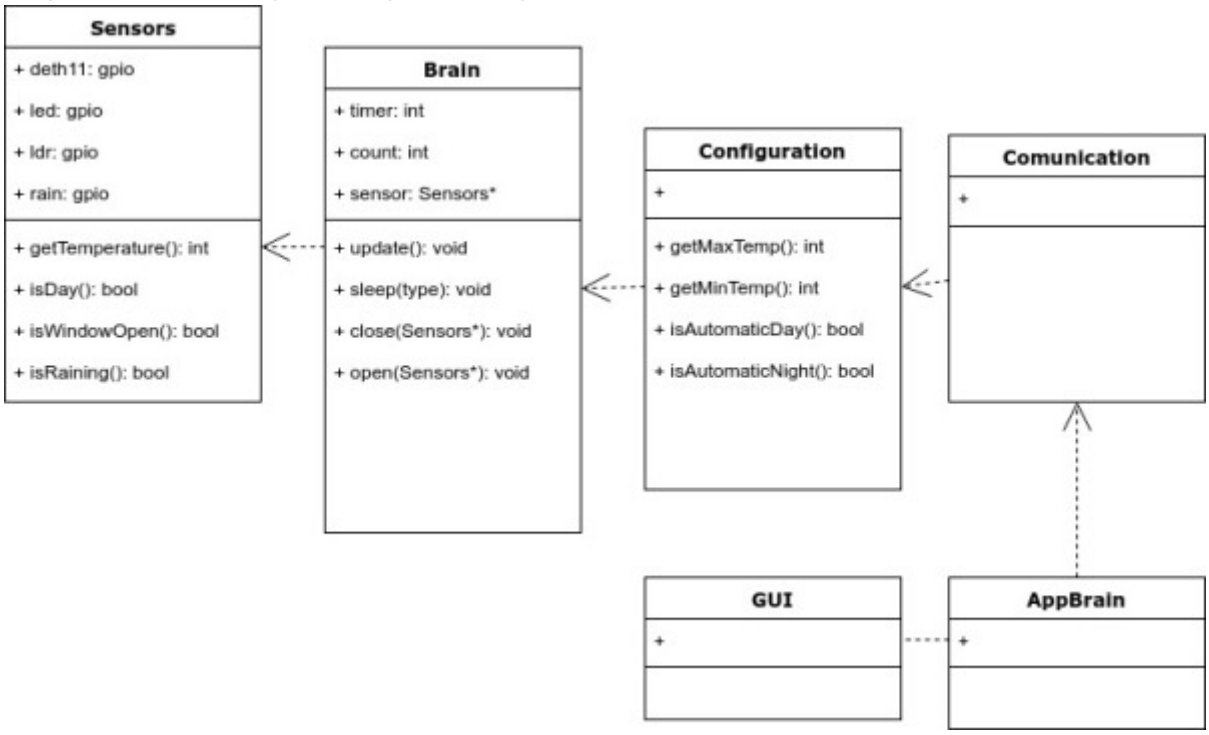


Figura 09: Diagrama de classes

A implementação será publicada em: https://github.com/karlajusten/automaticWindow

Software de Simulação

Foi desenvolvido um projeto em que apresenta a lógica esperada pelo Brain. O Brain é o decisor de ações para a janela, tais quaisdeverão ser tomadas a partir dos dados recebidos pelos sensores. Para verificar os sinais atuais dos sensores, foi criado um método chamado Update(), este atualiza os valores atuais de todos os sensores e verifica, também, as configurações selecionadas, como temperatura mínima, máxima e automático dia ou noite. Para fazer a decisão, é utilizado o método Act(), que, a partir dos estímulos, toma uma das 3 decisões: Abrir, fechar ou nenhuma. Para tratar os dados recebidos pelos sensores e as configurações do usuário, foram usadas variáveis para simulação, conseguindo assim controle para tratar e testar todas as situações possíveis. Dessa maneira, ainda não há dependências com o Raspeberry pi, podendo ser compilado e executado em qualquer pc.

Para realizar os testes, foi utilizado o framework de automatização de testes do Google, chamado de GTest (21). Contudo, tivemos dificuldades em agrupar todos os testes para serem executados sequencialmente e este relatório foi gerado a partir de uma série de compilações e execuções manualmente.

O código está disponível em: https://github.com/karlajusten/automaticWindow_Simulation

Com o auxilio do make file desenvolvido, para compilar o projeto, basta utilizar o comando: $ make; para executá-lo: ./janela; para compilar e executar os testes: $ make test

Para a simulação, foi criado o método setConditionstoSimulate(), nele enviamos uma tupla do forma:

setConditionstoSimulate({Estado da janela}, {Presença de chuva}, {Temperatura}, {Presença de luz}).

Threads

Foi criado um repositório no git para solucionar o problema da chuva. Nosso aplicação principal checa todos os sensores periodicamente, contudo, caso uma chuva ocorra, o ideal seria que a janela fechasse automaticamente. Para isso, foram criadas duas threads, uma thread principal na qual checa periodicamente todos os sensores e uma outra thread que observa o valor do sensor de chuva constantemente. O repositório com a implementação pode ser encontrado aqui: https://github.com/Luanes/automaticWindowThread

Software integrado e seus testes

Criamos duas baterias de simulação, cada uma com suas pré condições. Cada simulação possuí uma tabela para referência como gabarito, no qual possuí as pré condições da janela e a resposta desejada na coluna "Ação esperada" e a uma última coluna com a "resposta da simulação."

- Para as seguintes simulações é esperado que a janela comece aberta.

Janela	Chuva	Temperatura	Dia/noite	Temperatura máxima configurada	Temperatura Máxima configurada	Ação esperada	Ação que a simulação tomou
Aberta	Sem chuva	25	Dia	17	24	Nenhuma	Nenhuma

Aberta	Sem chuva	24	Dia	17	24	Nenhuma	Nenhuma
Aberta	Sem chuva	25	Dia	17	24	Nenhuma	Nenhuma
Aberta	Sem chuva	23	Dia	17	24	Nenhuma	Nenhuma
Aberta	Chuva	21	Dia	17	24	Fechar	Fechar
Fechada	Chuva	20	Dia	17	24	Nenhuma	Nenhuma
Fechada	Sem chuva	19	Noite	17	24	Nenhuma	Nenhuma
Fechada	Sem chuva	18	Noite	17	24	Nenhuma	Nenhuma
Fechada	Chuva	17	Noite	17	24	Nenhuma	Nenhuma
Fechada	Sem chuva	16	Noite	17	24	Nenhuma	Nenhuma
Fechada	Sem chuva	25	Noite	17	24	Abrir	Abrir

- Para as seguintes simulações é esperado que a janela comece aberta e que seu modo automático noite esteja desativado.

Janela	Chuva	Teperatura	Dia/noite	Temperatura mínima configurada	Temperatura Máxima configurada	Ação esperada	Ação que a simulação tomou
Aberta	Com chuva	20	Dia	17	24	Fechar	Fechar
Fechada	Com chuva	21	Dia	17	24	Nenhuma	Nenhuma
Fechada	Sem chuva	15	Dia	17	24	Nenhuma	Nenhuma
Fechada	Sem chuva	26	Dia	17	24	Abrir	Abrir
Aberta	Com chuva	13	Dia	17	24	Fechar	Fechar
Fechada	Sem chuva	22	Dia	17	24	Nenhuma	Nenhuma
Fechada	Sem chuva	27	Noite	17	24	Nenhuma	Nenhuma
Fechada	Com chuva	24	Noite	17	24	Nenhuma	Nenhuma
Fechada	Sem chuva	30	Noite	17	24	Nenhuma	Nenhuma
Fechada	Sem chuva	29	Noite	17	24	Nenhuma	Nenhuma
Fechada	Sem chuva	13	Noite	17	24	Nenhuma	Nenhuma

Testes realizados com a integração dos sensores ao brain:

- Para os seguintes testes, o sistema está configurado com a janela comece aberta e que seu modo automático noite esteja ativado e automático dia esteja ativado.

Janela	Chuva	Teperatura	Dia/noite	Temperatura mínima configurada	Temperatura Máxima configurada	Ação esperada	Ação que o teste tomou
Fechada	Sem chuva	26	Dia	17	24	Abrir	Abrir
Aberta	Sem chuva	26	Dia	17	24	Nenhuma	Nenhuma
Aberta	Com chuva	26	Dia	17	24	Fechar	Fechar
Fechada	Com chuva	26	Dia	17	24	Nenhuma	Nenhuma
Fechada	Sem chuva	26	Noite	17	24	Abrir	Abrir
Aberta	Sem chuva	26	Noite	17	24	Nenhuma	Nenhuma
Aberta	Com chuva	26	Noite	17	24	Fechar	Fechar
Fechada	Com chuva	26	Noite	17	24	Nenhuma	Nenhuma

- Para os seguintes testes, o sistema está configurado com a janela comece fechada e que seu modo automático noite esteja desativado e automático dia esteja desativado.

Janela	Chuva	Teperatura	Dia/noite	Temperatura mínima configurada	Temperatura Máxima configurada	Ação esperada	Ação que o teste tomou
Fechada	Sem chuva	26	Dia	17	24	Nenhuma	Nenhuma
Aberta	Sem chuva	26	Dia	17	24	Nenhuma	Nenhuma
Aberta	Com chuva	26	Dia	17	24	Nenhuma	Nenhuma
Fechada	Com chuva	26	Dia	17	24	Nenhuma	Nenhuma
Fechada	Sem chuva	26	Noite	17	24	Nenhuma	Nenhuma
Aberta	Sem chuva	26	Noite	17	24	Nenhuma	Nenhuma
Aberta	Com chuva	26	Noite	17	24	Nenhuma	Nenhuma
Fechada	Com chuva	26	Noite	17	24	Nenhuma	Nenhuma

- Para os seguintes testes, o sistema está configurado com a janela comece fechada e que seu modo automático noite esteja desativado e automático dia esteja ativado.

Janela	Chuva	Teperatura	Dia/noite	Temperatura mínima configurada	Temperatura Máxima configurada	Ação esperada	Ação que o teste tomou
Fechada	Sem chuva	26	Dia	17	24	Abrir	Abrir
Aberta	Sem chuva	26	Dia	17	24	Nenhuma	Nenhuma
Aberta	Com chuva	26	Dia	17	24	Fechar	Fechar
Fechada	Com chuva	26	Dia	17	24	Nenhuma	Nenhuma
Fechada	Sem chuva	26	Noite	17	24	Nenhuma	Nenhuma
Aberta	Sem chuva	26	Noite	17	24	Nenhuma	Nenhuma

Aberta	Com chuva	26	Noite	17	24	Nenhuma	Nenhuma
Fechada	Com chuva	26	Noite	17	24	Nenhuma	Nenhuma

- Para os seguintes testes, o sistema está configurado com a janela comece fechada e que seu modo automático noite esteja ativado e automático dia esteja desativado.

Janela	Chuva	Teperatura	Dia/noite	Temperatura mínima configurada	Temperatura Máxima configurada	Ação esperada	Ação que o teste tomou
Fechada	Sem chuva	26	Dia	17	24	Abrir	Abrir
Aberta	Sem chuva	26	Dia	17	24	Nenhuma	Nenhuma
Aberta	Com chuva	26	Dia	17	24	Fechar	Fechar
Fechada	Com chuva	26	Dia	17	24	Nenhuma	Nenhuma
Fechada	Sem chuva	26	Noite	17	24	Abrir	Abrir
Aberta	Sem chuva	26	Noite	17	24	Nenhuma	Nenhuma
Aberta	Com chuva	26	Noite	17	24	Fechar	Fechar
Fechada	Com chuva	26	Noite	17	24	Nenhuma	Nenhuma

Teste Realizado com a Janela

Janela	Chuva	Temperatura	Dia/noite	Temperatura máxima configurada	Temperatura Máxima configurada	Ação esperada	Ação que a simulação tomou
Fechada	Sem chuva	23	Dia	15	20	Nenhuma	Abrir
Aberta	Sem chuva	23	Dia	15	25	Nenhuma	Nenhuma
Aberta	Sem chuva	23	Dia	15	20	Nenhuma	Nenhuma
Aberta	Sem chuva	23	Dia	15	26	Nenhuma	Nenhuma
Aberta	Chuva	23	Dia	15	26	Fechar	Fechar
Fechada	Chuva	23	Dia	15	26	Nenhuma	Nenhuma
Fechada	Sem chuva	23	Noite	18	26	Nenhuma	Nenhuma
Fechada	Sem chuva	23	Noite	15	23	Nenhuma	Nenhuma
Fechada	Chuva	23	Noite	15	29	Nenhuma	Nenhuma
Fechada	Sem chuva	23	Noite	25	29	Nenhuma	Nenhuma

Fechada	Sem chuva	23	Noite	15	20	Abrir	Abrir

Aplicativo Mobile - “Controle Remoto - Janelas"

Para desenvolver o aplicativo mobile, será utilizada a ferramenta gratuita online App Inventor (http://appinventor.mit.edu/explore/).

Links que ensinam fazer comunicação wifi entre app inventor e raspberry pi:

1. http://www.instructables.com/id/Control-Raspberry-Pi-GPIO-Using-an-App/
2. https://docs.google.com/document/d/1jfNO_d8im8NI_RrlsidU1p2KWQw2GjLPsIPH7aERujw/edit#heading=h.4kmrqfqo1lyb

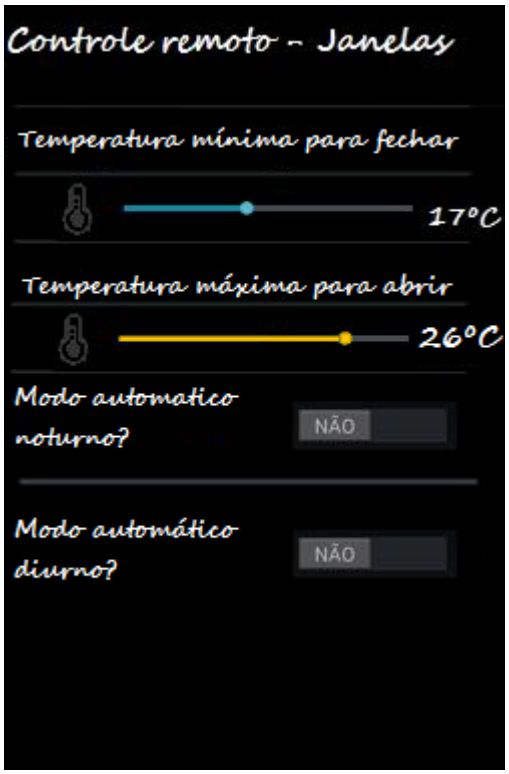


Figura 10: Protótipo de tela do app

Requisitos:

1. Aplicativo android 4.4 ou superior;
2. Permitir definição temperatura mínima do comodo, para quando estiver abaixo dela a janela fechará;
3. Permitir definição temperatura máxima do comodo, para quando estiver acima dela a janela abrirá;
4. Permitir desligar/ligar o sistema de automatização da janela durante dia;
5. Permitir desligar/ligar o sistema de automatização da janela durante noite;
6. Quando ambas opções de automatização durante dia e noite estiverem desligado, equivale a modo manual;

Produção do aplicativo : AppInventor

Como dito anteriormente, a produção do aplicativo mobile, foi utilizado a ferramenta "App Inventor". O App

Inventor, uma ferramenta de código-aberto, desenvolvido na Google sendo mantido pelo Instituto de Tecnologia de Massachusetts (MIT), que auxilia no desenvolvimento de aplicativos para dispositivos Android com suporte para design de interface e para a programação funcional do aplicativo. App Inventor é um ambiente de programação em blocos, como por exemplo Scratch e Snap!.

Para criar o seu aplicativo, basta você possuir uma conta google e fazer o login na plataforma. A tela inicial da plataforma é bem simples e bem intuitiva, nela você encontra um projeto já criado chamado "HelloPurr". Para criar um projeto novo, existe um botão no canto superior esquerdo chamado "Start new project". Ao clicar no botão, uma caixa de texto pedindo um nome irá surgir, e, após preencher o nome, o projeto vai ser incluído na sua lista de projetos (juntos com o "HelloPurr"). Para abrir o projeto recém criado, basta clicar apenas uma vez e começar a configurar o aplicativo.

Seguindo o protótipo sugerido na etapa de planejamento, incluímos peças para o layout. As peças adicionadas foram:

- 6 Labels para identificação sendo eles:

- Um para o título do aplicativo, chamado de "Automatic Window;
- Um para identificar o slider que defina a temperatura mínima, chamado de "Minimum Close Temperature";
- Um para identificar a caixa de texto no qual exibe a temperatura demarcada pelo slider de temperatura mínima.
- Um para identificar o slider que defina a temperatura máxima, chamado de "Maximum Close Temperatura";
- Um para identificar a caixa de texto no qual exibe a temperatura demarcada pelo slider de temperatura máxima.
- Um para identificar as checkbox de quando o sistema deve estar no modo automático, chamado de "Automatic Mode";

- 2 Sliders para o usuário escolher a posição;

- Um deles para controlar a temperatura mínima e o outro para controlar a temperatura máxima.

- 2 Checkboxes;

- Cada um deles referente à habilitação/desabilitação do sistema automático referente ao dia e a noite.

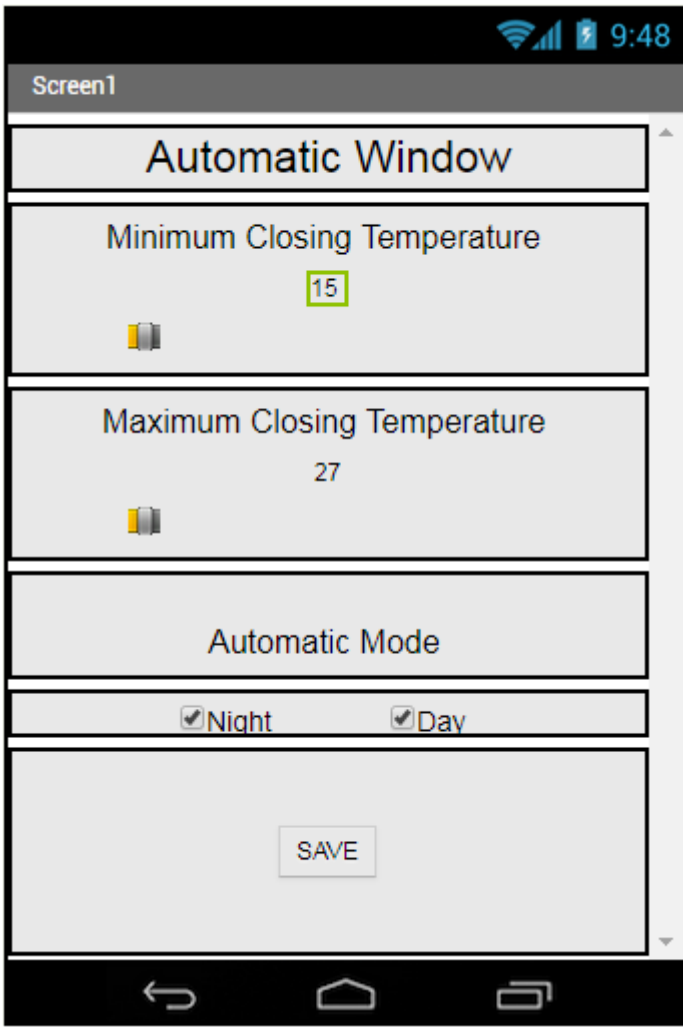


Figura 11: Exibição do aplicativo na plataforma.

Para realizar a computação, o App Inventor utiliza programação em blocos. Para acessar a tela de programação em blocos, basta ir ao canto superior direito da tela de design e clicar no botão "Blocks". Nele você encontrará diversos blocos no qual você poderá encaixar para fazer a computação do seu aplicativo. Assim que você abre a tela, ela começa em branco. Todas as entidades criadas ficam ocultas inicialmente, você só precisa incluir elas caso elas precisam fazer alguma ação. No nosso caso, temos os sliders e o label no qual representa o valor do slider. Os blocos utilizados nesse projeto são responsáveis por:

- 2 Conjunto de blocos para controlar a temperatura mínima e máxima exibida na caixa de texto das respectivas temperaturas;
- 1 Bloco para disparar a mensagem para salvar a configuração no servidor Tiny Webdb
- 2 Blocos para as checkboxes responsável pela configuração de automático dia/noite.

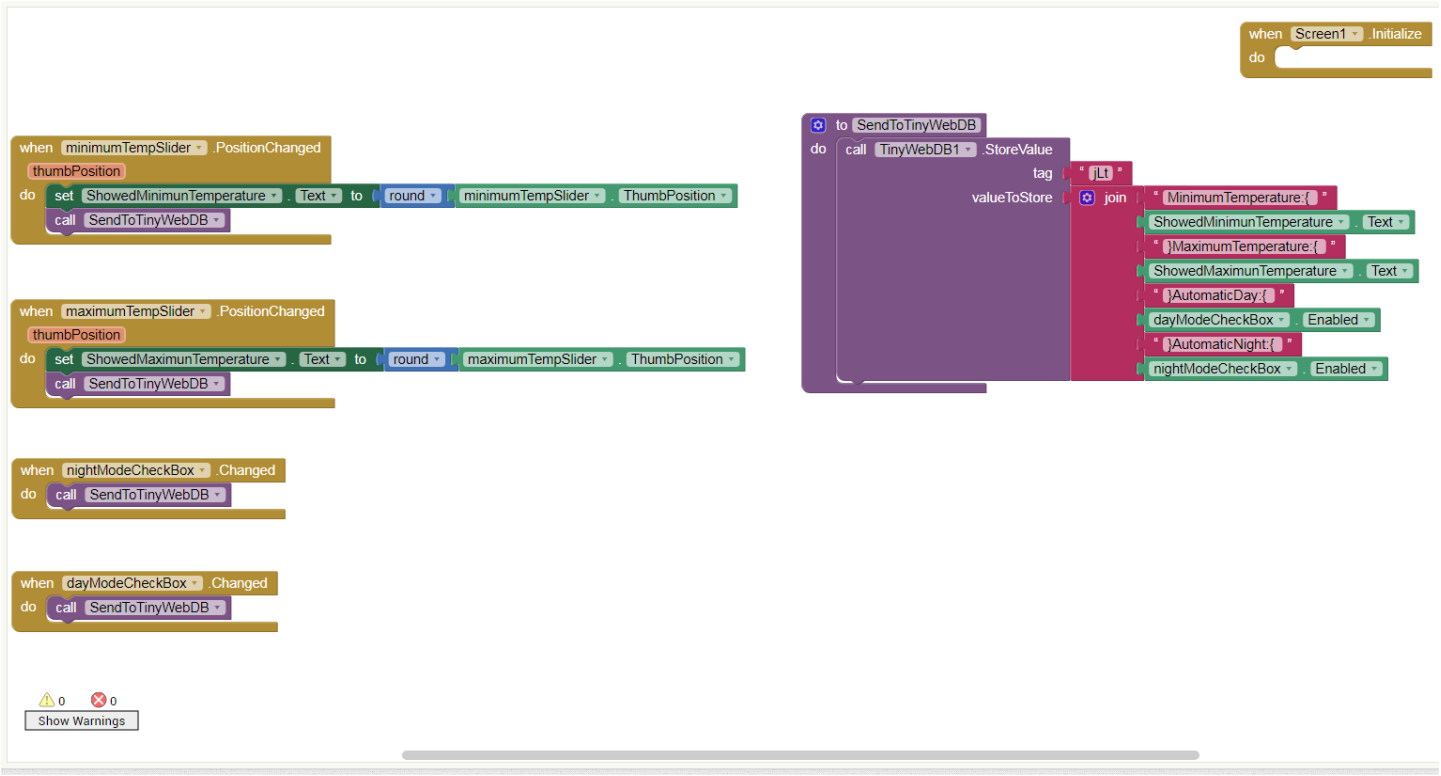


Figura 12: Blocos utilizados para o sistema.

No final de cada conjunto de blocos, é chamado o bloco "SendToTinyWebDB". Nele que é realizado o envio da mensagem para o servidor que irá armazenar as configurações do usuário.
Sobre o Tiny WebDB, é uma ferramenta bem simples para enviar e armazenar mensagens. Ele possui 2 serviços:
Um para armazenar uma string, você precisa definir uma "tag" que irá acessar aquela string. Observando pela Figura 12, utilizamos a tag "jLt" para armazenar os valores do bloco "SendToTinyDB". A leitura dessa informação será feito pela própria aplicação no raspberryPi, falaremos sobre isso mais à frente.

Inicialmente foi colocado um boão save, para enviar os dados para TinyWebDB, mas foi considerado melhor enviar as informções sempre que uma mudança for feita. Tornado o botão disnecessário.

A comunicação do alicativo com a base de dados TinyWebDB é apresentado em:
https://www.youtube.com/watch?v=Fxk-CxclCJM

Você pode visualizar a interface do aplicativo no celular a qualquer momento, basta você ir até o menu "Build", selecuinar "AI companion" e então você receberá um código para ser carregado no aplicativo(disponível na google store) MIT AI2 Companion. Todas as alterações feitas na aplicação web são imediatamente atualizadas no aplicativo conectado com o smart phone. Ao conectar ou realizar o download do aplicativo, ele ficará com a seguinte interface:

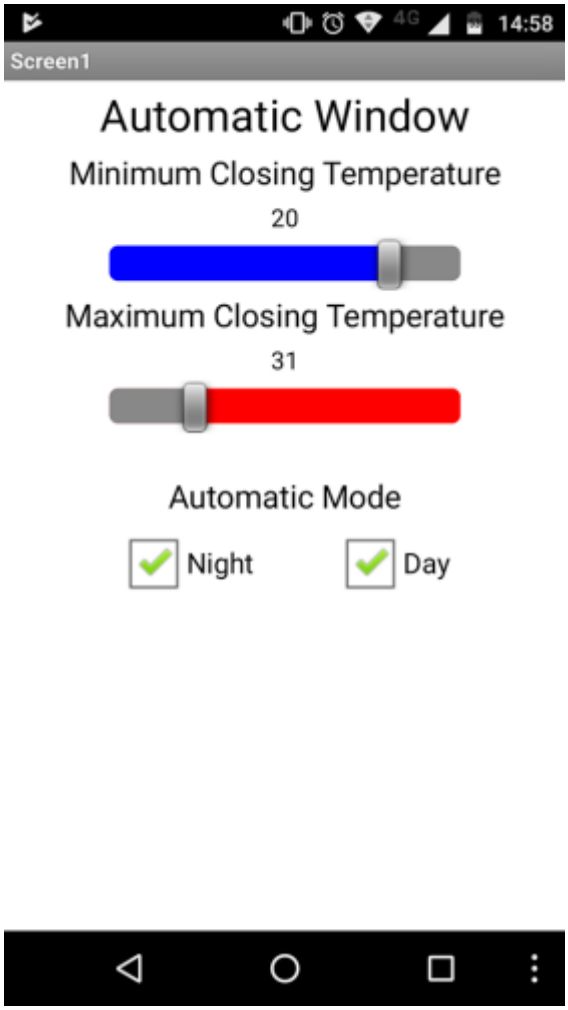


Figura 13: Aplicativo sendo executado em um dispositivo mobile.

Comunicação Tiny WebDB - Raspberry Pi

Para realizar a comunicação, foi utilizado a biblioteca CURL. Com essa bibilioteca, foi possível acessar o serviço do Tiny Webdb. Configuramos para que o raspberry envia uma mensagem para o tiny webdb, com o payload sendo a tag "jLt", tag na qual foi utilizada para armazenar as configurações da aplicação. A resposta do servidor se dá como um conjunto de strings. Visto que o tamanho da resposta é fixo, dado uma tag específica, basta consultar as posições respectivas de cada informação (temperatura mínima, temperatura máxima, automaticDay, automaticNight) para extrai-la.

A implementação da comunicação está no git(https://github.com/karlajusten/automaticWindow), nos arquivos "Connection.h" e "Connection.cpp". A parte de extração de informações se dá no método "retrieve()" do arquivo "configuration.cpp"

Parte Energética

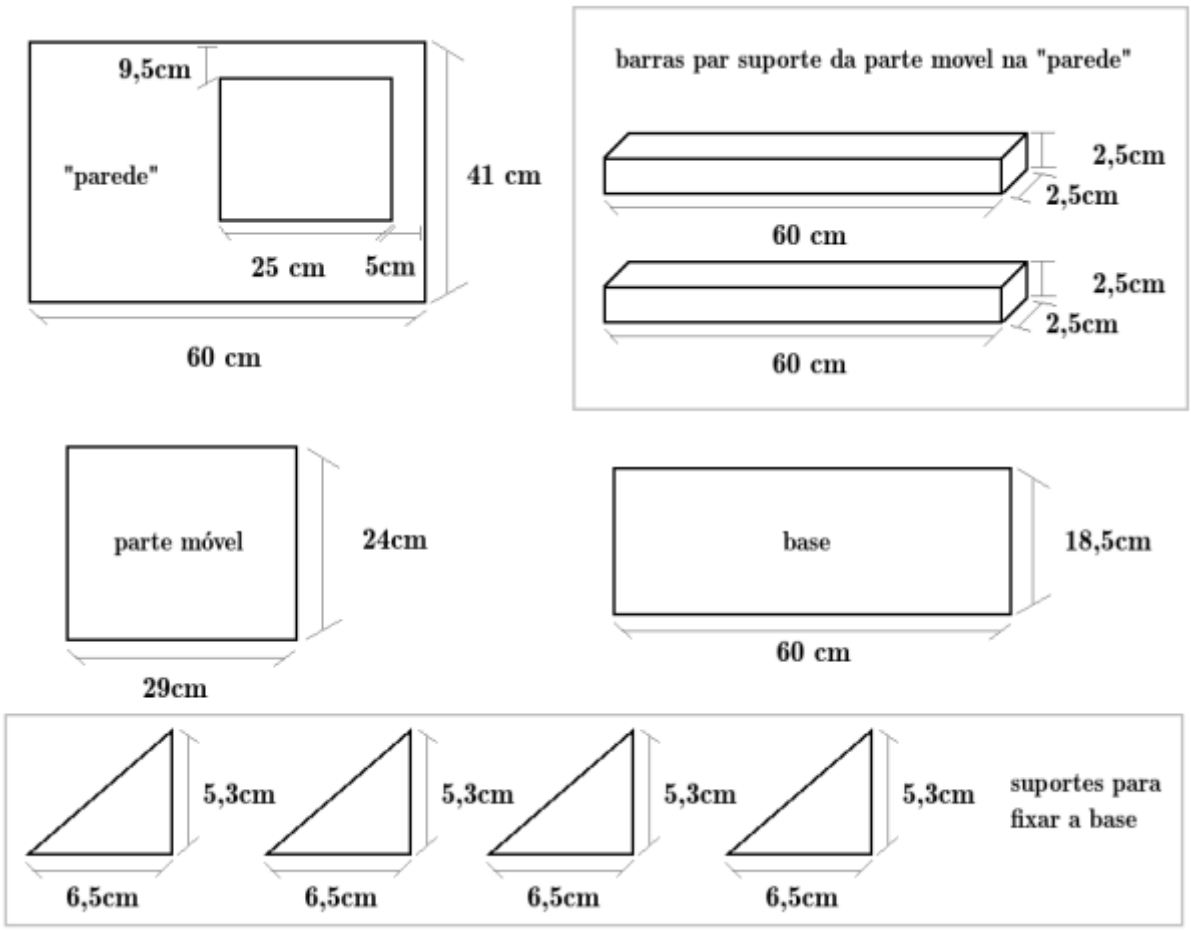
Com a intenção de identificar o quanto é o consumo energético do projeto aplicado, será utilizado um dispositivo, disponibilizado pelo LISHA, que ao conectar no Raspberry Pi é registrado o consumo de energia.

Assim será possivel verificar a viabilidade financeira e estrutural da utilização da energia solar como fonte de energia.

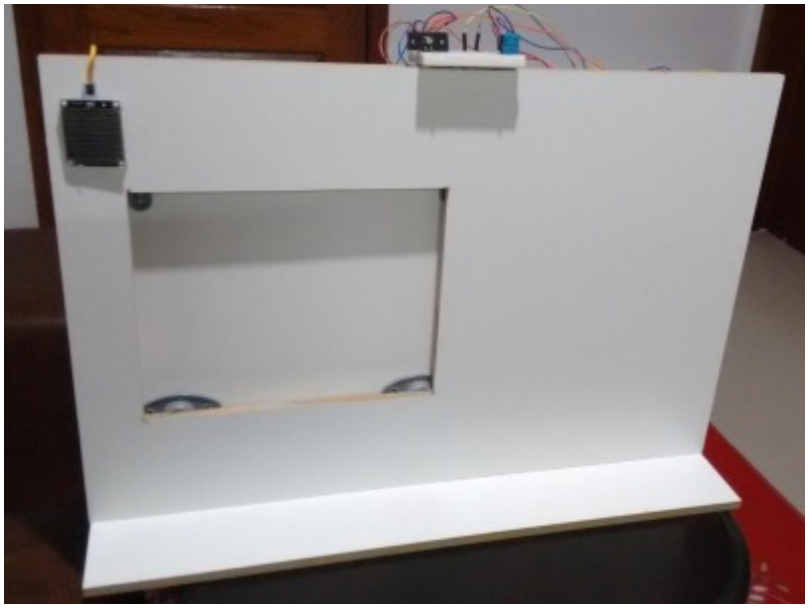
Como exemplo, em 17 é apresentado uma abordagem para gerenciamento de consumo energético em sistemas embarcados.

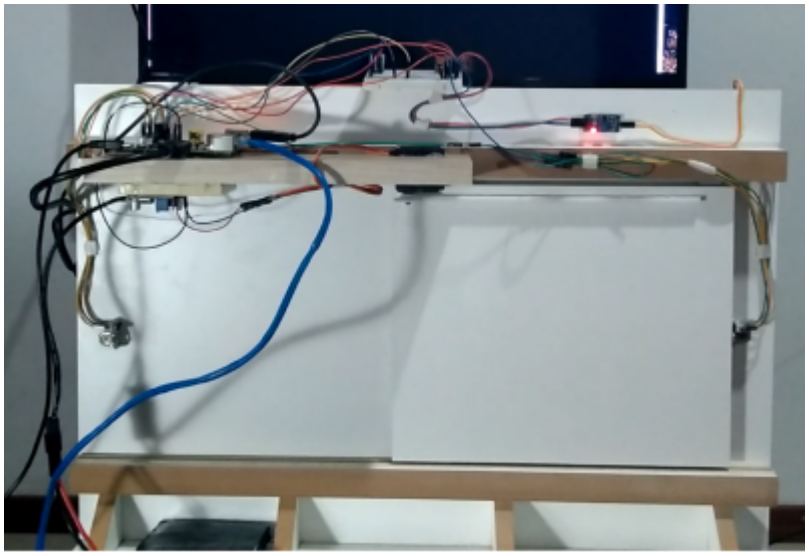
Projeto de construção do Protótipo de Janela

Para construir o protótipo de janela foram utilizadas pedaços chapas de compensados, como apresentado na figura a seguir:



Para fixar uma madeira na outra foi utilizado super cola de cianoacrilato.





Orçamento

Orçamento do quanto seria necessário para comprar tudo o que fosse necessário, excto mouse USB, teclado USB e Monitor HDMI. Os preços apresntados nos produtos marcados por * realmente forma pagos pelo valor indicado para realizar este projeto. Os demais valores são uma estimação com base no mercado.

Nome	Quantidade	Preço Unitário (R$)	Total	Fornecedor(a)
Madeira*	1	15	15	Karla
Serviço de corte das madeiras*	1	15	15	Karla
Super Cola	1	16,8	16,8	Karla
batedor (“L”)	2	2,5	5	Karla
Interruptor 3 pinos 1A 125V AC	2	3,86	3,72	Karla
Conjunto Cremalheira e Engrenagem*	1	24,9	24,9	Karla
Servo Motor TowerPro SG 5010	1	34,9	34,9	Karla
Cabos macho-macho	pct	16,9	16,9	Karla
Cabos macho-femea	pct	13,52	13,52	Lisha
Regulador de Tensão LM2596*	1	9,9	9,9	Karla
Bateria 7V	1	30,00	30	Karla
Raspberry pi 1 B	1	69,0	69	Lisha
Cabo de Rede	1	6,59	6,59	Karla
Sensor de temperatura e Humidade DHT11 de 4 pinos	1	12,9	12,9	Lisha
Sensor de Chuva - YL83*	1	12,9	12,9	Karla
Sensor de Claridade - Resistor dependente da luz (LDR)	1	1,4	1,4	Lisha
Resistores	2	1,9	1,9	Lisha
		Total	280,81	

Bibliography

1. PACHECO, A. P. R. et al. O ciclo PDCA na gestão do conhecimento: uma abordagem sistêmica. PPGEGC – Universidade Federal de Santa Catarina – Programa de Pós Graduação em Engenharia e Gestão do Conhecimento., v. 2, 2012. url: http://isssbrasil.usp.br/artigos/ana.pdf
2. Prescient - Adaptive Context-Aware Monitoring and Control System (Intel Embedded Systems Challenge 2016 - https://epos.lisha.ufsc.br/Prescient Acessado em Setembro de 2017.
3. Pratley & Partners, Automated windows - http://www.ljpratley.co.uk/blog/why-go-automated-with-your-windows
4. Fast Homes Vídeo - https://www.youtube.com/watch?v=2jb3nOibwCA
5. Anderson Casement Window - http://www.quadomated.com/technology/automation/anderson-casement-window-24-vdc-electric-motor-automation/ Acessado em Setembro de 2017.
6. Fakro - https://www.youtube.com/watch?v=KrgsaPVTV3I Acessado em Setembro de 2017.
7. Blackouts automatizados - http://www.makeuseof.com/tag/automate-window-blinds/ Acessado em Setembro de 2017.
8. Automatic Cared Home for Aging People - https://www.kics.or.kr/storage/paper/event/2015_winter2014/publish/14C-34.pdf Acessado em Setembro de 2017.
9. Designing a home of the future - http://ieeexplore.ieee.org/document/1012340/#full-text-section Acessado em Setembro de 2017.
10. Trabalho de conclusão de curso, JANELA AUTOMATIZADA PARA SMART HOUSES - http://repositorio.uniceub.br/bitstream/123456789/3381/3/20516450.pdf Publicado em Dezembro de 2010.
11. Comparative study of the indoor air quality of naturally ventilated and air-conditioned bedrooms of residential buildings in Singapore: http://www.sciencedirect.com/science/article/pii/S0360132304000356#aep-section-id13 Acessado em Setembro de 2017.
12. Aproveitamento de energia solar - https://books.google.com.br/books?hl=pt-BR&lr=&id=ZJOAJNe8n5oC&oi=fnd&pg=PP10&dq=Automatic+Home+casement&ots=ZXDQ5fW6kK&sig=NdOun75pRV4ljaCWehK-9mw5VcU#v=onepage&q&f=false Acessado em Setembro de 2017.
13. Hydraulic System Automation - Hydraulic System Automation (ESL 2015/2) Acessado em Setembro de 2017.
14. Especificação do EPOS - https://epos.lisha.ufsc.br/EPOSMote+III Acessado em Setembro de 2017.
15. Sensor de chuva - https://www.filipeflop.com/blog/sensor-de-chuva-yl-83/ Acessado em Setembro de 2017.
16. Sensor de chuva - https://www.filipeflop.com/produto/sensor-de-chuva/ Acessado em Setembro de 2017.
17. A Comprehensive Approach to Power Management in Embedded Systems http://journals.sagepub.com/doi/10.1155/2011/807091
18. http://www.uugear.com/portfolio/using-light-sensor-module-with-raspberry-pi/
19. http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-the-raspberry-pi/
20. https://learn.sparkfun.com/tutorials/raspberry-gpio/c-wiringpi-api
21. https://github.com/google/googletest
22. http://www.uugear.com/portfolio/dht11-humidity-temperature-sensor-module/
23. https://www.sunfounder.com/learn/sensor-kit-v2-0-for-raspberry-pi-b-plus/lesson-14-rain-detection-module-sensor-kit-v2-0-for-b-plus.html

24. https://stackoverflow.com/questions/16040128/hook-up-raspberry-pi-via-ethernet-to-laptop-without-router/35529971#35529971