Sistemas Embutidos
CC4040

Casa Experta

Realizado por:
Cláudia Maia (up201905492)
Eduardo Marim (up201700176)
Gabriel Alves (up201709532)
João Pedro Silva (up202004063)

Conteúdo

1 Introdução

Os sistemas embutidos são sistemas informáticos concebidos para executar tarefas específicas dentro de um sistema ou dispositivo maior. São frequentemente concebidos para serem de baixa potência e têm capacidades limitadas, pois são altamente optimizados para a sua tarefa específica.

Os Sistemas Ciber-Físicos (CPS) são um tipo de sistema incorporado que combina componentes físicos com capacidades computacionais, tais como sensores e actuadores. Estes sistemas podem monitorizar e controlar processos físicos em tempo real, permitindo uma interacção perfeita entre o mundo físico e o digital.

As casas inteligentes são um exemplo de CPS que utilizam sistemas incorporados para automatizar e optimizar vários aspectos da vida doméstica, tais como iluminação, aquecimento, e segurança. Estes sistemas utilizam frequentemente sensores para detectar alterações no ambiente e ajustar as definições em conformidade, e podem ser controlados remotamente através de um smartphone ou outro dispositivo. Ao integrar CPS em casas inteligentes, os proprietários de casas podem alcançar maior comodidade, eficiência energética e segurança.

2 Especificação e Planeamento

2.1 Requisitos

- Implementar 5 divisões com luz automática.
- Acender todas as luzes em baixa luminosidade, executando em menos de 2s.
- Controlar com smartphone, executando em menos de 5s.
- Mostrar a temperatura da casa.
- Acender a luz, com movimento na divisão.
- Reproduzir música na casa a partir do smartphone.

2.1.1 Requisitos Funcionais

- Segurança: O sistema deve estar equipado com medidas de segurança, como proteção por senha, criptografia e autenticação para impedir o acesso não autorizado.
- Acesso Remoto: O sistema deve permitir que os utilizadores acessem e controlem remotamente os vários recursos da sua casa inteligente.
- Eficiência energética: O sistema deve ser projetado para economizar energia, desligando automaticamente luzes e outros dispositivos quando não estiverem em uso e ajustando as configurações de temperatura com base na ocupação e hora do dia.
- Expansibilidade: O sistema deve ser projetado para permitir expansão futura e integração com outros dispositivos domésticos inteligentes, como, fechaduras, sensores ou câmaras de segurança, para proporcionar uma experiência doméstica inteligente mais abrangente.
- Personalização: O sistema deve permitir que os utilizadores personalizem vários settings e preferências, como o nível de escurecimento das luzes ou o volume da música.

- Suporte a múltiplos utilizadores: O sistema deve suportar múltiplos utilizadores, cada um com suas próprias configurações e preferências personalizadas.
- Manipulação de erros: O sistema deve ser projetado para lidar com erros e exceções de forma adequada, com mensagens de erro e notificações apropriadas.
- Privacidade de dados: O sistema deve ser projetado levando em consideração a privacidade de dados, garantindo que os dados do utilizador sejam protegidos e não partilhados com terceiros não autorizados.
- Controlo manual: O sistema deve permitir que os utilizadores controlem manualmente as luzes e outros dispositivos, seja por meio de interruptores físicos ou por meio da aplicação para smartphone, caso prefiram não usar os controlos automáticos.
- Alimentação elétrica: O sistema deve ser projetado para conectar-se à alimentação elétrica principal, em vez de depender de baterias, para garantir que o sistema possa operar continuamente, sem a necessidade de substituição ou recarga de bateria.
- Estética: O sistema deve ser projetado com uma estética limpa e moderna que combine com o design da casa e não seja visualmente intrusiva ou desatente. Isso inclui a aparência física dos dispositivos, a interface do utilizador da aplicação para smartphone e a experiência do utilizador geral do sistema.

2.1.2 Requisitos Não Funcionais

- Rapidez: O sistema deve responder rapidamente e de forma confiável aos inputs do utilizador, com atraso ou lag mínimos.
- Escalabilidade: O sistema deve ser projetado para suportar um grande número de dispositivos e utilizadores e deve ser capaz de escalar para cima ou para baixo conforme necessário.
- Segurança: O sistema deve ser projetado com segurança em mente, com medidas em vigor para proteger os dados do utilizador e impedir o acesso não autorizado.
- Compatibilidade: O sistema deve ser compatível com uma ampla gama de dispositivos e plataformas, para garantir que possa ser facilmente integrado a outros sistemas domésticos inteligentes.
- Usabilidade: O sistema deve ser fácil de usar e navegar, com interfaces claras e intuitivas que sejam acessíveis a utilizadores de todos os níveis de habilidade.
- Confiabilidade: O sistema deve ser projetado para operar de forma confiável em uma variedade de condições, com tempo de inatividade ou falhas de sistema mínimos.
- Manutenibilidade: O sistema deve ser projetado com manutenibilidade em mente, com componentes de fácil substituição e uma arquitetura modular que facilite reparos e atualizações.
- Acessibilidade: O sistema deve ser projetado para ser acessível a utilizadores com deficiências, com recursos como texto-voz e suporte a dispositivos de entrada alternativos.
- Conformidade: O sistema deve cumprir as regulamentações e normas relevantes, como as leis de proteção de dados e normas específicas da indústria para sistemas domésticos inteligentes.

- Desempenho sob carga: O sistema deve ser projetado para funcionar bem sob carga pesada, como durante os horários de pico de uso, para garantir que possa lidar com um grande número de pedidos simultâneos sem diminuir ou falhar.
- Extensibilidade: O sistema deve ser projetado para permitir a adição de novos recursos e capacidades, como suporte a novos dispositivos ou serviços sem a necessidade de grandes reformulações.

2.2 Modelo

2.2.1 Diagrama de Blocos

A figura 1 descreve a interação entre os vários elementos de uma maneira geral, abstraida do hardware em concreto, onde observamos que pela applicação do smartphone conseguimos realizar as várias tarefas propostas, como ler a temperatura da casa, reproduzir música, e gerir a iluminação da casa. Quanto às lâmpadas, existe uma lógica maior associada, por tem de ter em conta tanto a presença de pessoas, como a luminusidade, isto automaticamente, pois também podemos ligar/desligar manualmente.

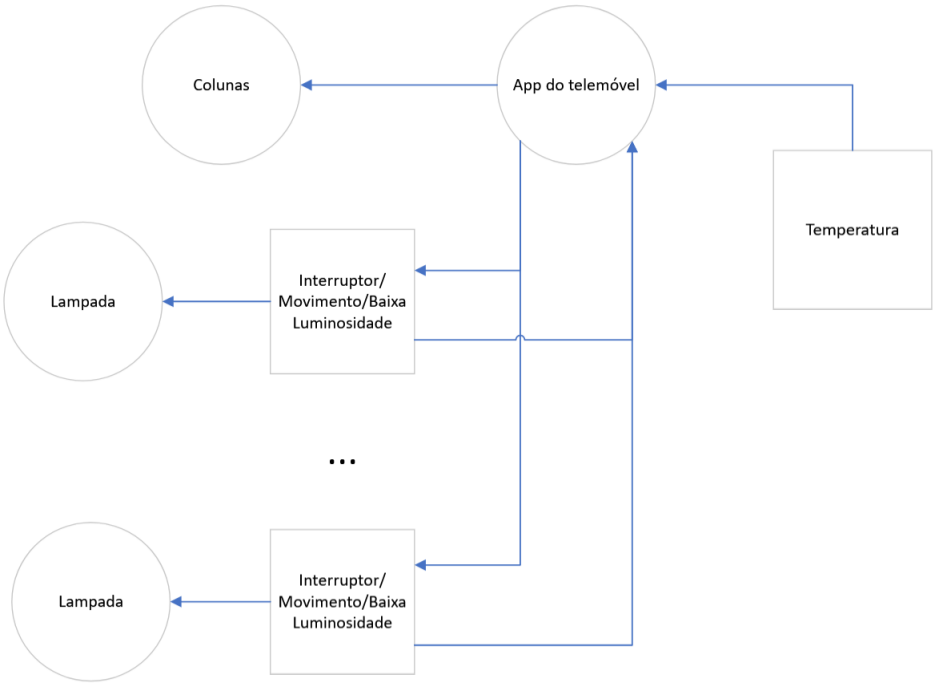

Figura 1: Diagrama geral do sistema

2.2.2 Diagrama de Hardware

Na figura 2 já temos uma visão mais perto do hardware, onde existe um Raspberry Pi, que age como servidor para os múltiplos Arduino, e smartphones e também interage com as colunas. Para cada

Arduino temos ligado um sensor de movimento, um interruptor, e a lâmpada associada. No caso do Módulo 1, temos acrescido o sensor de temperatura e luminosidade.

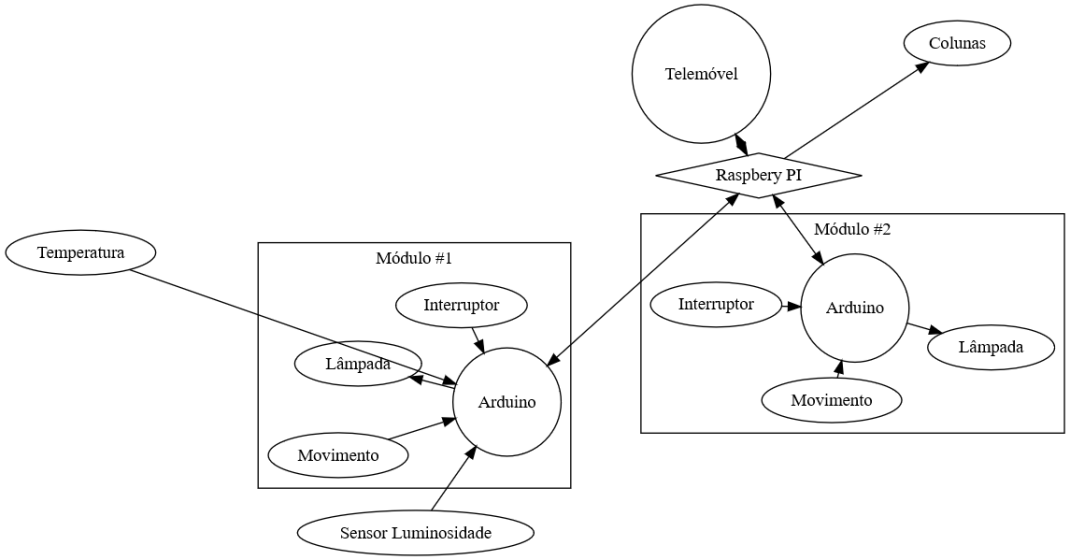

Figura 2: Diagrama da integração do hardware

2.2.3 Máquina de estados

Por último, a figura 3 é o desenho da máquina de estados da lâmpada. Os possíveis estados apesar de ser ligada ou desligada, temos mais alguns para representar o motivo do mesmo, isto é, o motivo de estar ligada, para gerir assim a precedência dos acontecimentos. A escolha foi a presença ser a menos prioritária, passando por luminosidade, e por fim manualmente, sendo o manual o mais prioritário.

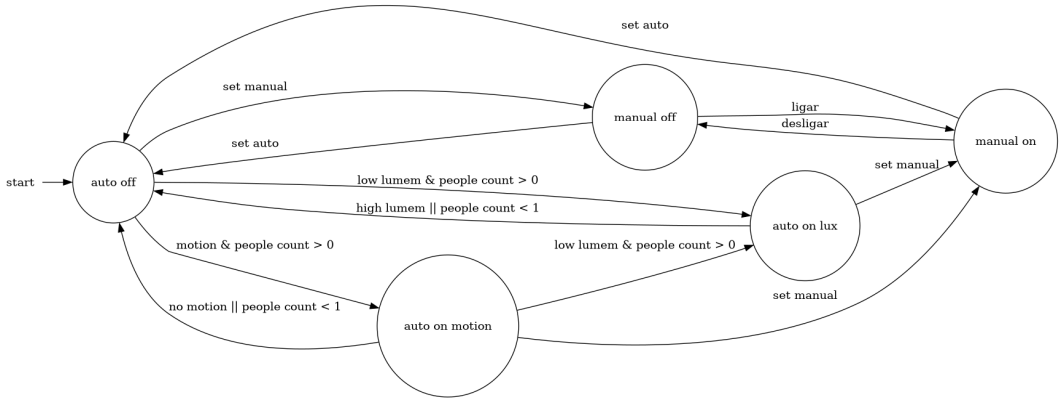

Figura 3: Máquina de estados da lâmpada

2.3 Lista de material

Materiais	Custos	Quantidade	Link
Arduino	20,52	5	Available: here
Tomada para RPI	9,47	1	Available: here
Suporte 8xAA Conector PP3	1,48	2	Available: here
Breadboard BB400	±6	5	
Botões	±1	5	
Light Diode Resistor Sensor GL5528	±1	5	
Temperature-Humidity Sensor DHT11	4	5	Available: here
Sensor de Movimento Mini PIR	5,10	5	Available: here
Infrared Emitter and Detector Pair	±1	2	
LEDs	± 0,20	5	
Raspberry Pi 3 ou 4	±40	1	
SD Card 32 GB	±4	1	
WiFi Adapt RPi	±10	1	
Custo total:	257,53		

Tabela 1: Lista de materiais e custos

2.4 Formulário de Requerimentos

Name: Casa Esperta

Purpose: Desenvolvimento de um protótipo com o objetivo de automação das casas

Inputs: Temperatura; movimento; interruptores (físicos e virtuais); luminosidade

Outputs: Lâmpadas; colunas (musica); temperatura (APP)

Functions: Acender as luzes com baixa luminosidade/ movimentação/ comandado; colocar musica a tocar; informar o estado das lampadas; informar a temperatura da divisão

Performance: Acender as luzes(todas) em baixa luminosidade em menos de 2s; Controlar com smartphone em menos de 5s

Manufacturing cost: 300€

Power: 20 W

Physical size/weight: 1 Kg

3 Implementação

A arquitetura deste projeto é baseada à volta do Rasbperry Pi como peça central e cérebro, uma vez que, lá será guardado o estado dos sensores, tal como se uma dada lâmpada está ligada ou desligada, a temperatura da casa ou mesmo a luminosidade desta. Este é o funcionamento de um MQTT Broker [2], é uma possível implementação para integrar os vários dispositivos na gestão da casa [1].

Os Arduinos vão comunicar com o MQTT Broker por internet sem fios previamente instalado na casa, e reagir às ações, tal como, ligar uma luz, interligando com o mundo ciberfísico.

Os sensores e os atuadores são a parte física que recebe os comandos dos Arduinos, sendo estes vindos do Raspberry Pi, também o Arduino lê o estado dos sensores e reporta ao Raspberry Pi.

A interação do smartphone com a smart home é através do Raspberry Pi, o MQTT Broker tem o estado de todos os sensores guardados, como também oferece a possibilidade de alterar o estado destes, como o caso de alterar o estado de uma lâmpada.

Em geral, vai tudo funcionar à volta do Raspberry Pi, mais específico, do MQTT, pois oferece uma integração simples e flexível.

Visto que as divisões da casa iriam ter as mesmas funcionalidades, optamos por colocar um Arduino em cada um delas com os respetivos sensores e atuadores. Para todos eles foi utilizada a mesma lógica, onde inicialmente foram configurados todos os pinos a serem utilizados, definidas as configurações do servidor mqtt e ainda a configuração do Wi-fi, dado que seria a forma como os Arduinos iriam comunicar com o Raspberry.

```
void setup() {
  pinMode(LED, OUTPUT);
  pinMode(LDR, INPUT);
  pinMode(PIR, INPUT);
  pinMode(BTNTGL, INPUT);
  #ifdef ENTRANCE
  pinMode(HOUSEIN, INPUT);
  pinMode(HOUSEOUT, INPUT);
  #endif
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
```

Relativamente à configuração da ligação Wi-fi, foi apenas necessário definir o modo da ligação e ainda iniciar a comunicação com as respetivas credenciais.

```
void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());
```

```
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

Uma vez realizadas todas as configurações necessárias, era então ora de passar à parte do código que iria estar a correr constantemente. Então a primeira coisa a fazer era verificar se a conexão se encontrava ativa. Caso não estivesse, era necessária a implementação dev um mecanismo que fosse responsável por restabelecer a conexão. Em seguida, optamos por criar um mecanismo que faça com que as leituras de temperatura e luminosidade não estejam a ser enviadas constantemente para o servidor a correr no Raspberry. Com isto, a frequênca com que estes valores eram lidos e enviados dependia do valor escolhido mas no caso optamos por atribuir um delay dev 3 segundos. Já os valores das leituras dos sensores de presença e de gestão dos botões fisicos não iriam ter delay nenhum uma vez que era requerido que o sistema fosse responsivo. O mesmo para a funcionalidade que indica a presença ou não de presenças na divisão.

```
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  unsigned long now = millis();
  if (now - lastMsg > PUBLISH_DELAY_MS) {
    lastMsg = now;
    publish_ldr();
    publish_dht();
  }
  publish_pir();
  handle_switch();
  #ifdef ENTRANCE
  publish_people();
  #endif
}
```

Para a leitura do ldr, tivemos de efetuar a leitura do mesmo e em seguida criar e publicar o topico indicando o valor lido. Uma vez criado o tópico era necessário proceder à publicação do mesmo.

```
void publish_ldr() {
  int ldr_val = 1024 - analogRead(LDR);
  char ldr_pub[MSG_BUFFER_SIZE];
  char topic[MSG_BUFFER_SIZE];

  snprintf(ldr_pub, MSG_BUFFER_SIZE, "%d", ldr_val);
  snprintf(topic, MSG_BUFFER_SIZE, "rooms/%d/light_sensor", ROOMID);

  client.publish(topic, ldr_pub);
  Serial.print("ldr: ");
  Serial.println(ldr_val);
}
```

Para a leitura do sensor de temperatura e humidade o procedimento é muito parecido onde temos dev efetuar a leitura do mesmo e em seguida criar, no caso, os tópicos uma vez que 1 deles continha o valor da temperatura e o outro da humidade. Nota só para o facto de termos criado uma lógica para determinar o sensor em uso para assim reaproveitarmos o código. Então em divisões que o

sensor utilizado seja o DHT a leitura é feita através da função *dht.read()*, caso se tratasse do DHT22 através da função *dht.read2()*.

```
void publish_dht() {
  #ifdef DHT11
  byte temperature_val = 0;
  byte humidity_val = 0;
  #endif
  #ifdef DHT22
  float temperature_val = 0;
  float humidity_val = 0;
  #endif
  char topicT[MSG_BUFFER_SIZE];
  char topicH[MSG_BUFFER_SIZE];
  snprintf(topicT, MSG_BUFFER_SIZE, "rooms/%d/temperature", ROOMID);
  snprintf(topicH, MSG_BUFFER_SIZE, "rooms/%d/humidity", ROOMID);

  char temperature_pub[MSG_BUFFER_SIZE];
  char humidity_pub[MSG_BUFFER_SIZE];

  int err = SimpleDHTErrSuccess;

  #ifdef DHT11
  if ((err = dht.read(DHT, &temperature_val, &humidity_val, NULL)) != SimpleDHTErrSuccess) {
    Serial.print("Read DHT11 failed.");
    return;
  }
  #endif
  #ifdef DHT22
  if ((err = dht.read2(DHT, &temperature_val, &humidity_val, NULL)) != SimpleDHTErrSuccess) {
    Serial.print("Read DHT22 failed.");
    return;
  }
  #endif

  snprintf(temperature_pub, MSG_BUFFER_SIZE, "%d", (int)temperature_val);
  snprintf(humidity_pub, MSG_BUFFER_SIZE, "%d", (int)humidity_val);

  client.publish(topicT, temperature_pub);
  client.publish(topicH, humidity_pub);

  Serial.print("temp: ");
  Serial.println(temperature_val);
  Serial.print("hum: ");
  Serial.println(humidity_val);
}
```

Para a leitura do sensor de movimento, após a leitura do valor do mesmo, verificavamos se o valor era diferente do anterior. Caso fosse era armazenado esse valor como o próximo para iteração e enviado o tópico quer fosse detetado movimento ou não.

```
void publish_pir() {
  char topic[MSG_BUFFER_SIZE];
  snprintf(topic, MSG_BUFFER_SIZE, "rooms/%d/presence_sensor", ROOMID);

  int pir_val = digitalRead(PIR);

  if (pir_val != pir_val_last) {
    pir_val_last = pir_val;

    if (pir_val) {
```

```
      client.publish(topic, "true");
    } else {
      client.publish(topic, "false");
    }

    Serial.print("pir: ");
    Serial.println(pir_val);
  }
}
```

Para a gestão dos switches, optamos por efectuar a sua leitura e sempre que este for ativo é enviado um tópico para o servidor e esse será o responsável por devcidir se o led deve ser ligado ou desligado.

```
void handle_switch() {
  bool btntgl = !digitalRead(BTNTGL);
  char topic[MSG_BUFFER_SIZE];
  snprintf(topic, MSG_BUFFER_SIZE, "rooms/%d/switch_toggle", ROOMID);

  if (btntgl != btntgl_last) {
    btntgl_last = btntgl;
    if (btntgl) {
      Serial.println("Switch_toggle");
      client.publish(topic, "true");
    }
  }
}
```

Para efetuar a gestão de pessoas na casa, tinhamos dois sensores, um correspondente à entrada e outro à saída. Mediante o sensor que fosse lido , era criado e publicado o tópico correspondente.

```
#ifdef ENTRANCE
void publish_people() {
  bool housein = !digitalRead(HOUSEIN);
  bool houseout = !digitalRead(HOUSEOUT);

  if (housein != housein_last) {
    housein_last = housein;
    if (housein) {
      Serial.println("Add_person");
      client.publish("house/people/add", "true");
    }
  }

  if (houseout != houseout_last) {
    houseout_last = houseout;
    if (houseout) {
      Serial.println("Remove_person");
      client.publish("house/people/remove", "true");
    }
  }
}
#endif
```

Para finalizar o código dos Arduinos, criamos ainda um método que iria receber informação do servidor. No caso era um payload que caso tivesse um ´t´ no caracter inicial o led seria ativo, senão o led iria ser desligado.

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message_arrived_[");
  Serial.print(topic);
```

```
  Serial.print("] ");
  for (unsigned int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  // Switch on the LED if an 1 was received as first character
  if ((char)payload[0] == 't') {
    digitalWrite(LED, HIGH);
  } else {
    digitalWrite(LED, LOW);
  }
}
```

4 Conclusão

Em conclusão, o nosso objetivo inicial era criar um módulo de Arduino para cada uma das cinco divisões. No entanto, devido à falta de sensores e LEDs, só foi possível desenvolver quatro módulos completos. Além disso, um dos módulos apresentou um problema com o sensor de movimento, mas bastava apenas substituí-lo por um sensor funcional. Embora tenhamos enfrentado essas limitações, foi evidente que poderíamos ter alcançado o nosso objetivo de criar os cinco módulos se tivéssemos mais materiais disponíveis. Já que, com exceção da divisão que exigia botões para contar o número de pessoas que entram e saem de casa, todos os outros módulos eram idênticos e facilmente escaláveis. Por fim, planeávamos usar sensores infravermelhos para detectar a presença de pessoas dentro de casa, mas devido à falta de material adequado, tivemos que recorrer ao uso de botões para realizar essa contagem. Assim, podemos verificar na figura 4 cada um dos modúlos e a casa esperta montada e em funcionamento, tal como foi demonstrado em aula.

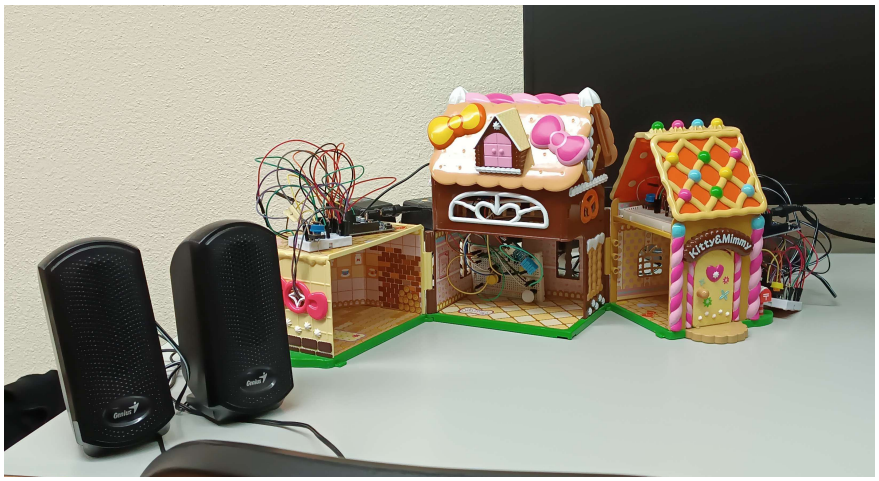

Figura 4: Casa Esperta

5 Link para o repositório do GitHub

https://github.com/Goloso98/SE

Referências

[1] Hans Boef. Control your smart home the open source way, Apr 2021.

[2] The standard for iot messaging.