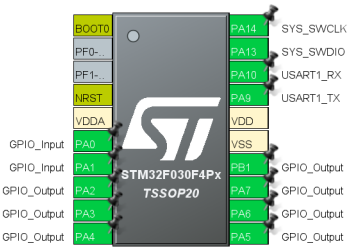
1 Configuração da Greenpill

Utilizando a STM32CUBEIDE definimos 6 portas como GPIOOUTPUT da Greenpill e 2 portas como GPIOINPUT,estas portas tinham como finalidade realizar enviar as entradas do circuito lógico(GPIOOUTPUT) e receber as saídas do circuito (GPIOINPUT).Além disso duas portas foram usadas para realizar comunicação com um periférico,UARTTX,UARTRX.



1.1 Configuração da UART

A UART foi utilizada para realizar o envio da equação lógica para o dispositivo bluetooth(HM-10).O bound rate utilizado nessa comunicação foi de 9600.Na comunicação se utilizou a UART em "Polling mode"para transmitir os dados da greenpill,isto é,ela parava o processo principal e só retornava após a conclusão da tarefa,que no caso era transmitir uma string. Inicialização da UART:

```
huart1.Instance = USART1;
  huart1.Init.BaudRate = 9600;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    Error_Handler();
}
```

1.2 O módulo bluetooth

O módulo utilizado foi o HM-10,ele é um módulo bluetooth 4.0,do tipo BLE.Ele é slave do nosso dispositivo e atua recebendo a saída da equação lógica e enviando para o telefone.

1.3 Desenvolvimento da aplicação mobile:App Inventor

Usamos no projeto o MIT App Inventor para desenvolver a aplicação que vai mostrar a equação lógica .O App Inventor permite o desenvolvimento de aplicações de maneira muito simples,pois não é necessário ter domínio sobre desenvolvimento de aplicativos para criar o app,apenas utilizamos blocos para possibilitar o desenvolvimento da aplicação.

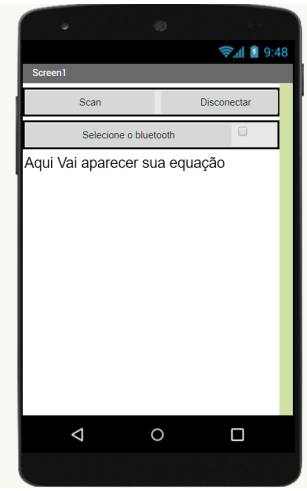


Figura 1: Modelo do app no app inventor

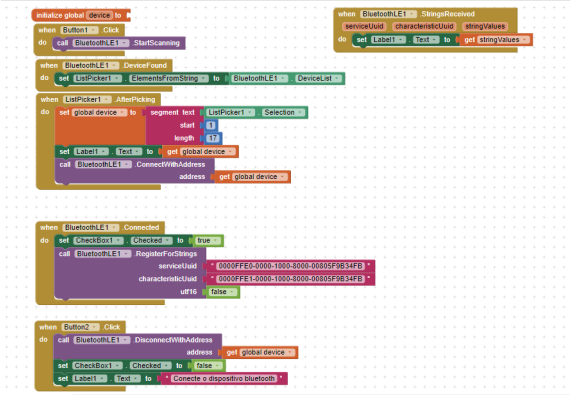


Figura 2: Modelo do app no app inventor

É preciso deixar de aviso,a quem precisar trabalhar com módulo bluetooth que os tipo BLE não tem uma biblioteca,no APP INVENTOR, tão fácil de usar quanto o convecional,é sempre necessário utilizar determinados números que caracterizam a comunicação,para envio e escrita.Para realizar essa identificação podemos usar o app nRF Connect,que retorna as ids de cada serviço.Esse app também é útil para monitorar o que esta sendo enviado e recebido pelo módulo,as desvantagens de seu uso é sua intromissão no uso de outras aplicações,então se deixado em segundo plano ele conecta com o

bluetooth mesmo sem permissão para isso.

2 Formação da tabela verdade

O processo de formação da tabela verdade é feito pela seguinte função:

```
void tabelaverdade(int tabela[],int numerodeentradas){
                int i;
                int *w;
                int value;
            for(i =0;i<power(2,numerodeentradas);i++){
                w = &i;
                for(int j=0;j<=numerodeentradas-1;j++){
                        value=ReadBit(w,j);
                        digitalWrite(correspondencia[j],value);
                }
                tabela[i]=digitalRead(14);
            }

}
```

A variável i é responsável por ser a variável de laço principal ela varia de 0 até $2^{numerodeentradas}$ − 1,onde a variável numerodeentradas representa o número de entradas do circuito lógico.Dentro do laço principal usamos um pointeiro para armazenar o endereço de i.O laço secundário se encontra dentro do principal,a variável j varia de 0 até numerode entradas -1 o que corresponde exatamente ao número de portas usadas como input do circuito.Dentro do laço secundário temos o uso da função ReadBit(),que retorna o valor o valor do bit na posição j,na posição de memória apontada por w.Em seguida escreve o valor 0 ou 1 na GPIO ,utilizamos nessa parte a equivalência do arduino,vamos explicar mais a frente como é feita.Ao sair do laço todas as entradas estão setadas com seus valores de 0 ou 1.Então,realizamos a leitura de outra GPIO,que atua como input e armazenamos o valor em um vetor que corresponde as saídas do circuito lógico.

2.1 Equivalência do arduino e vetor de correspondência

Para realizar a escrita e leitura das GPIOs utilizamos as funções desenvolvidas na prática de GPIO.Nesta prática realizamos uma correspondência entre arduino e as portas da greenpill,essa coresspondência foi utilizada na prática nas funções digitalWrite() e digitalRead().

Tabela de OUTPUT da Greenpill.

Pino do arduino	porta da GreenPill
1	PA2
0	PA3
16	PA4
13	PA5
12	PA6
11	PA7

Tabela de INPUT da Greenpill

Pino do arduino	porta da GreenPill
14	PA0
15	PA1

O vetor de correspondência é um vetor que carrega as correspondências do arduino,ele utiliza a seguinte organização:

A(MSB)	B	C	D	E(LSB)
PA6	PA5	PA4	PA3	PA2

Um exemplo de set das saídas é dado a seguir:

1. No laço i = 8.Em binário é 1000.
2. W aponta para i.
3. j começa em 0 e realiza a leitura do bit na posição 0 do espaço de memória apontado por W.Que é zero
4. Setamos a GPIO de PA2 para 0.
5. Realizamos a leitura novamente e para j =1 e j=2 a leitura do bit igual a zero.
6. Para j=3,o bit lido tem valor 1.Então setamos o GPIO de PA4 para 1.
7. Para j=4 temos GPIO de PA5 para 0.
8. lemos no GPIO da porta PA0,o valor da saída do circuito para entrada 01000.