3.1 Considerações Iniciais

Nesse Capítulo encontram-se as implementações presentes no projeto tanto de *hardware* quanto de *software*. Na parte de *software*, é explicado o funcionamento dos módulos como um todo e por fim, o funcionamento detalhado de cada função utilizada. HARDWARE MISSING Também estão presentes os resultados obtidos, análises dos resultados e uma análise do projeto como um todo, levando em conta as dificuldades e limitações que surgiram durante o processo.

3.2 Projeto

O objetivo do projeto é desenvolver um sistema de irrigação autônomo que, através de imagens da área a ser irrigada e dados coletados da internet e de sensores, decida quais regiões precisam ser irrigadas e o quanto de água cada uma dessas regiões necessita.

O sistema de irrigação autônomo possui duas frentes igualmente importantes: O *software* que é o responsável por controlar o sistema e tomar a decisão de regar ou não a área e o *hardware*, que é onde o *software* irá ficar hospedado e também responsável pelas ações do sistema.

O *software* é dividido em duas partes, a parte de tomada de decisão, que foi desenvolvida na linguagem C++ e a parte responsável por ler os sensores e transmitir os valores obtidos, que foi desenvolvida na linguagem do Arduino, que é baseada na linguagem C.

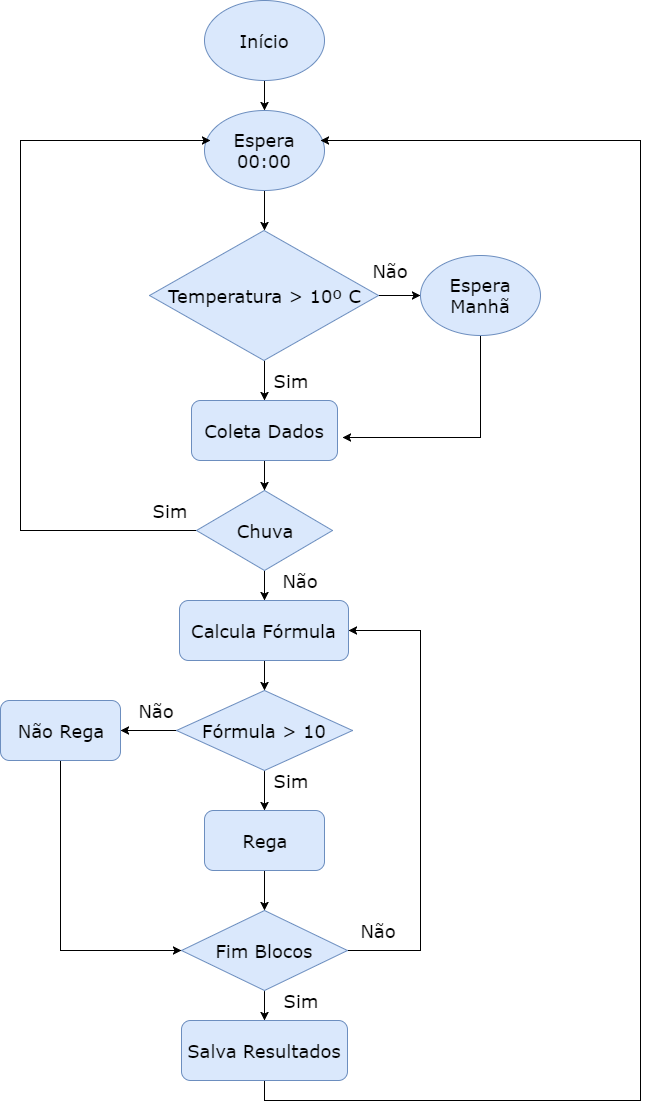
O *hardware* consiste na Raspbery Pi, hospedeira do programa responsável pela tomada de decisão, um Arduino, responsável por ler os valores dos sensores e outro Arduino responsável por controlar o motor que regula a quantidade de água usada para regar uma área.

3.2.1 Software

O programa principal, o que recebe os dados e a imagem e gera a decisão para cada bloco a ser irrigado foi dividido em quatro módulos: Núcleo, onde se encontra a função *main*, que comanda todo o programa; Processamento de Imagem, que é onde a imagem recebida pelo programa é processada a fim de extrair as informações necessárias; Tomada de Decisão, onde se encontram as funções responsáveis por obter os dados da *internet* e dos sensores, assim como a função que salva os resultados em um arquivo e Sprinkler, módulo que se encarrega de moldar o jato que vai irrigar o bloco selecionado.

3.2.1.1 Núcleo

No núcleo, todos os outros módulos são chamados através de suas funções. Seu fluxo principal está representado no fluxograma abaixo:



(Fonte: Elaborado pelo Autor)

O programa foi projetado como um laço que espera sempre o horário de regar as plantas, que nesse projeto foi definido como meia noite (00:00). Esse horário foi escolhido ########.

Caso a temperatura esteja muito baixa (abaixo dos 10º C), opta-se por regar as plantas de manhã, evitando que a água congele.

Definido o horário em que as plantas serão regadas, chama-se, então, as funções responsáveis por coletar os dados. Desses valores coletados, o primeiro a ser utilizado é o coeficiente de chuva. Ele representa a probabilidade de chuva contabilizando a previsão do tempo para o dia atual e os próximos dois (ensolarado, nublado, chuvoso, tempo claro e nevando) e quanto maior o seu valor, menor a necessidade de se regar as plantas. Se o número for muito baixo, significa que não há necessidade e então o laço volta para o começo, esperando pelo horário da próxima irrigação.

Após a verificação da possibilidade de chuva, vem o procedimento principal do programa, que é outro laço onde se faz o cálculo da fórmula que diz se o bloco (a imagem com a área total a ser irrigada é dividida em blocos, sendo que para cada um é calculada a fórmula para decidir se eles serão regados ou não) deve ser irrigado ou não. A fórmula leva em conta os valores coletados anteriormente, que são: Insolação, Velocidade do Vento, Umidade do Solo, Coeficiente de Chuva, Cor da Área e o Resultado do dia Anterior. Assim como o coeficiente da chuva, esse cálculo resulta num valor numérico entre 0 e 100 onde 100 é a necessidade máxima de água e de 10 até 0 significa que não é necessário regar aquele bloco.

Feito o cálculo da fórmula e decidido se o bloco deve ser irrigado ou não, faz-se o mesmo para os próximos até o fim dos blocos. Ao terminar o laço interior, os resultados são salvos num arquivo para serem utilizados no dia seguinte e então o programa volta a esperar pela meia noite.

3.2.1.2 Processamento de Imagens

O módulo do programa responsável pelo processamento de imagens é extremamente dependente da biblioteca OpenCV, portanto algumas etapas de seu fluxo são funções implementadas pela biblioteca. O fluxo completo deste módulo encontra-se no fluxograma, como pode ser visto na Figura XXX abaixo:

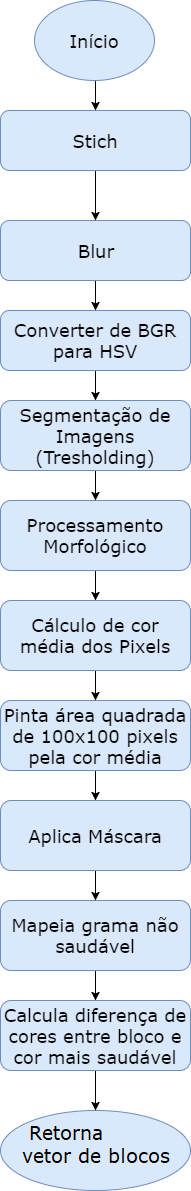


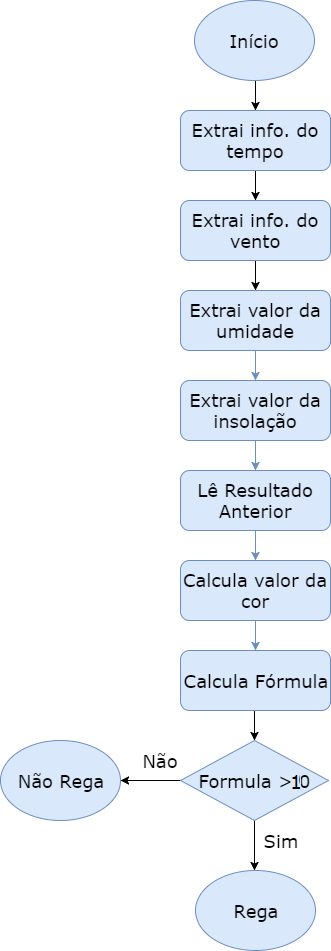
Figura XXX – Descrição do fluxo completo do módulo de processamento de imagens.

O processamento de imagens nesse projeto é uma sucessão de ações em cima das imagens iniciais, resultando em um vetor de blocos que representa as áreas que serão avaliadas para irrigação. Seu fluxo é simples e não possui nenhum laço de repetição ou rota alternativa, essa sequência de ações é chamada no início do programa e só é executada uma vez.

As imagens que chegam como parâmetro para o programa são juntadas através do método de *stitching* e a partir desse resultado são feitas as operações necessárias para extrair a informação desejada da imagem. O *blur*, conversão do espaço BGR (RGB na notação do OpenCV) para HSV, segmentação da imagem e processamento morfológico são todos métodos implementados pela biblioteca do OpenCV e são necessários para manter na imagem somente o que é desejado, que no caso desse projeto é a grama. A partir da imagem gerada por essas operações, são calculados a cor média dos pixels de bloco em bloco, sendo cada bloco um quadrado de lado de 100 *pixels*. Após esse cálculo, todos os pixels dentro da área onde foi calculada a média são “pintados” na cor média. Após essa transformação, é aplicada uma máscara para determinar quais blocos são considerados saudáveis e quais não são. Os blocos considerados não saudáveis, ou blocos que necessitam ser regados, são salvos num vetor que será mandado ao núcleo do programa onde será feito o processo de tomada de decisão.

3.2.1.3 Tomada de Decisão

Assim como no processamento de imagens, o fluxo de tomada de decisão é uma sequência de operações com a diferença que no final existem duas possibilidades: regar ou não regar o bloco. O fluxograma da tomada de decisão é:



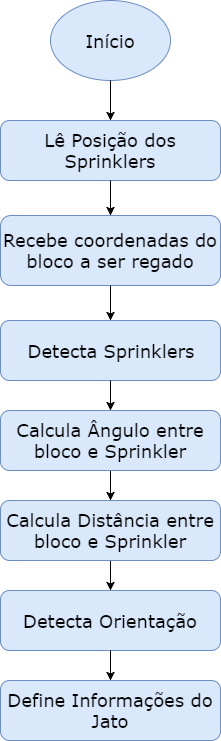
TEMPO <>CLIMA

As informações do tempo e do vento são retiradas da internet através de uma linha de comando, já as informações da umidade do solo e insolação são obtidas através dos respectivos sensores. O resultado anterior é lido de um arquivo onde todos os resultados são salvos, como mostrado no fluxograma do núcleo do programa. Por fim, o valor da cor é o módulo do vetor diferença entre a cor do bloco e a cor que é o valor inferior no intervalo que define se a grama precisa ser regada, ou seja, é o valor da grama menos saudável.

Após a obtenção de todos os valores necessários, faz-se o cálculo da fórmula para decidir se o bloco vai ser regado ou não. A partir da leitura do resultado anterior até o final, o fluxo é o mesmo que o fluxo do *loop* interior encontrado no Núcleo. Isso acontece pois para a tomada de decisão, não existe uma função que englobe todas as outras do mesmo módulo como acontece com o processamento de imagens, portanto, as funções são chamadas individualmente pelo núcleo.

3.2.1.4 Aspersor

O módulo responsável pelos sprinklers é o mais simples, sendo dois métodos os mais importantes: O método responsável por ler os sprinklers de um arquivo e o método que define os jatos. Esse último detecta o sprinkler adequado para o bloco que foi passado como parâmetro e chama todas as funções restantes do módulo: a do cálculo da distância entre sprinkler e módulo e a do cálculo do ângulo entre sprinkler e módulo. A orientação é determinada na detecção de qual sprinkler será utilizado.



(Fonte: Elaborado pelo autor)

3.3 Descrição das Atividades Realizadas

Como foi dito na seção anterior, o projeto possui duas frentes: o *software* e o *hardware*. Nesta sessão encontra-se a explicação para as implementações de cada módulo e HARDWARE.

3.3.1 Software

O software produzido para esse projeto é *open source* e encontra-se em <https://github.com/pedrosmv/TCC/> . O código referente à tomada de decisão, feito na linguagem C++, foi dividido em quatros arquivos, cada um relativo a um módulo e três headers, um para cada módulo menos o núcleo, que não possui função além da *main*.

3.3.1.1 Processamento de Imagem

3.3.1.1.1 Stitch

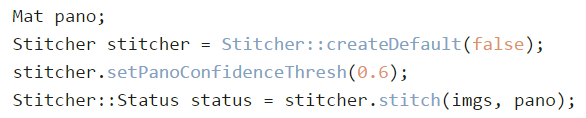
A primeira função do módulo de processamento de imagens é stitch:



Os seus argumentos, *argc* e *argv* são os mesmos recebidos pela *main* passados via linha de comando. O inteiro *argc* é um valor que indica a quantidade de argumentos que foram passados na chamada do programa. Já *argv* é um vetor do tipo *char* que contém os argumentos. É necessário atentar-se para o fato que *argv[0]* é o nome do programa, portanto os argumentos encontram-se nas posições posteriores.

O tipo de retorno da função, Mat, é um tipo definido pela biblioteca OpenCV, que representa um *array* de canal simples ou multicanal que pode armazenar imagens. No caso da função aqui citada o retorno é uma imagem, união das imagens passadas por argumento.

As imagens presentes no vetor *argv* são lidas e colocadas num vetor de imagens do tipo Mat chamado “imgs” e esse vetor será utilizado pelo método stitch do objeto sitcher, como é possível observar na imagem abaixo:



A imagem resultado, presente em pano, é o retorno da função.

Nesse projeto, durante o desenvolvimento, as imagens de entrada foram:



Após a chamada da função, o resultado foi:

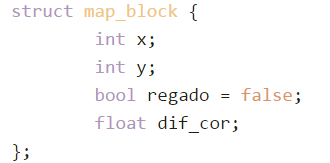


3.3.1.1.2 Image Processing



Essa função é o método principal do módulo de processamento de imagens. É ela que vai chamar todas as outras funções, exceto a stitch(), e extrair o vetor de blocos que será utilizado pelo núcleo do programa.

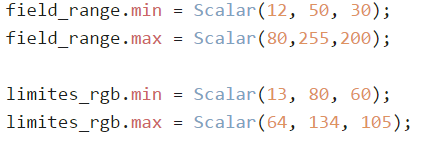
Esse vetor de blocos, que é o tipo do retorno da função é um vetor da *struct* map\_block, determinada no *header* do módulo, imageprocessing.h:



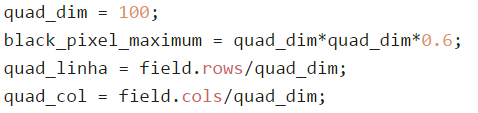
Nessa struct estão agrupados os valores: x, posição X do bloco no mapa de pixels; y, posição Y do bloco no mapa de pixels; regado, uma variável booleana que diz se o bloco foi regado ou não e dif\_cor, o valor da diferença entre a cor do bloco e a cor que é o valor inferior no intervalo que define se a grama precisa ser regada, ou seja, é o valor da grama menos saudável.

Como argumentos, image\_processing recebe a imagem gerada pela função stitch() e o endereço de dois ponteiros, max\_col representando o número máximo de colunas e max\_linha representando o número máximo de linhas. Essas variáveis são utilizadas no módulo do sprinkler e, portanto, seus valores são atribuídos no módulo de processamento de imagens, único local onde esse valor é acessível.

A primeira parte dessa função é definir o intervalo de cores que define a área de interesse da imagem, no nosso caso, o gramado (field\_range) e o intervalo em que o gramado necessita de água (limites\_rgb). Os valores usados nesse projeto foram definidos na monografia do Tiago.



Após a definição dos intervalos, são definidas algumas variáveis que são utilizadas entre todas as funções:



Essas variáveis estão relacionadas com a divisão da imagem em blocos, quad\_dim é a quantidade de pixels que formam um lado do bloco, black\_pixel\_maximum define a quantidade máxima de pixels que podem ser pretos dentro de um bloco e quad\_linha e quad\_col representam o número de blocos no eixo Y e X da imagem respectivamente.

O próximo passo da função é aplicar as operações necessárias para poder extrair da imagem as informações desejadas. A primeira parte utiliza apenas funções implementadas pelo OpenCV, resultando em uma imagem em preto e branco onde o branco é gramado e o preto é o que não deve ser considerado nas próximas operações.

A primeira dessas funções é a blur, usada para suavizar a imagem:



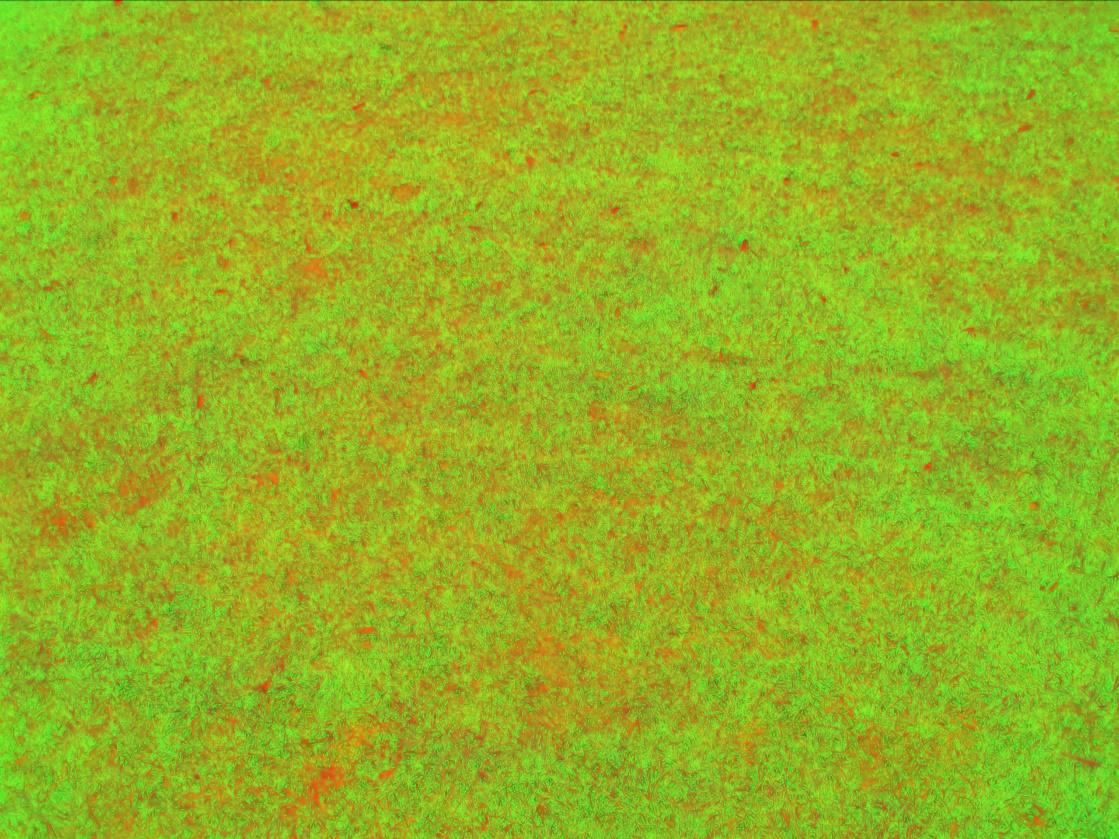
O resultado pode ser visto na figura abaixo:



Após suavizar a imagem, é feita a conversão do espaço de cores BGR (RGB na notação do OpenCV) para HSV:



O resultado pode ser visto na figura abaixo:



Com a imagem no espaço de cor HSV, é feita a segmentação na imagem, usando como intervalo de *treshold* o intervalo field\_range, que vai delimitar o que é gramado e o que não é:



Como na imagem utilizada para teste não há nada além de gramado, o resultado, que pode ser visto na figura X, não apresenta uma região em preto, apenas pequenos pontos, que serão corrigidos pelo processo morfológico:



O processo morfológico encarrega-se de eliminar as imperfeições do *tresholding*, resultando na figura Y.



(Segmentação)



(Processamento morfológico)

Após o processamento morfológico, a imagem está pronta para ser processada a fim de fornecer as informações desejadas. A partir da figura Y, é chamada a função:



Essa função vai calcular a média das cores dos pixels numa área de 100x100 pixels e pintar todos os pixels dentro dessa área da cor média. O resultado é uma imagem quadriculada, como na figura abaixo V:

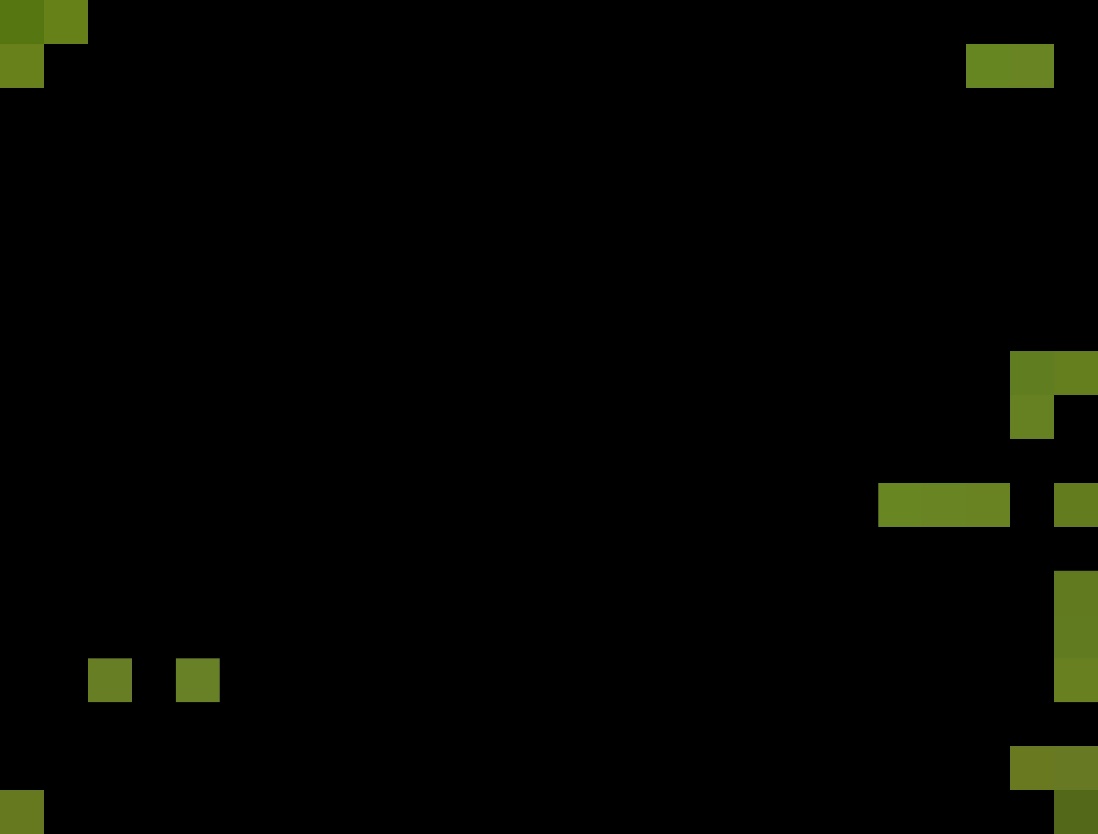


Nessa imagem, utilizando o intervalo determinado em limites\_rgb, será feita outra segmentação, a fim de gerar uma máscara que separará os blocos que devem ser regados daqueles que não devem ser regados.

A máscara é gerada pela função:



O resultado é uma imagem onde os blocos que devem ser regados estão em preto e os blocos considerados saudáveis estão em branco. Após copiar os pixels brancos para uma outra imagem, o resultado é a imagem abaixo Z:



Por fim, a função que gera o vetor de blocos que será retornado pela image\_processing é 

Essa função utiliza as figuras V e Z para dizer se o bloco está saudável ou não. Se o bloco é considerado não saudável, é criado um bloco do tipo map\_block que é salvo no vetor de blocos. Terminada a execução dessa função, termina a execução da image\_processing e o vetor de blocos é retornado.

3.3.1.1.3 Calculate Average Pixel Color



A função calculateAvgPxlColor é responsável por transformar a imagem que passou pelo processamento morfológico na imagem quadriculada vista na figura V.

Seus argumentos são: final\_field, imagem que passou pelo processamento morfológico; quad\_dim, dimensão do lado do quadrado que vai ser pintado com a cor média dos pixels dentro de sua área; quad\_linha representa o número de blocos no eixo Y da imagem; quad\_col representa o número de blocos no eixo X da imagem e black\_pixel\_maximum é o máximo de pixels que podem ser pretos dentro de um bloco. O retorno da função é um objeto do tipo Mat, classe do OpenCV usada para representar uma imagem.