

**Autores:**

Alexandre Machado – 2020100636

Miguel Gonçalves – 2020100302

Paulo Bernardino - 3807

**Tarefas concluídas a 100%:**

Neste momento temos a ligação do Raspberry Pi a transmitir vídeo, que iremos usar essa transmissão para produzir numa fonte externa de modo a conseguirmos visualizar a válvula em tempo real. Outra tarefa que temos praticamente completada, que não conseguimos completar a 100% porque depende de outros fatores que só iriam estar prontos com o resto dos sensores em condição funcionar é a fonte externa de visualização dos conteúdos presentes no projeto, essa fonte externa é uma aplicação em flutter que desenvolvemos que recebe as informações, como o valor da válvula, a transmissão da câmara e o resto dos sensores e o seu estado em tempo real.

**Tarefas agendadas:**

Para tarefas agendadas temos várias já em movimentação. Uma delas é a utilização de uma BreadBoard para ligação de vários sensores que ao mesmo tempo necessitam que seja implementado no código utilizado para o controlo do Raspberry Pi. Outra tarefa é a obtenção dos sensores que iremos usar:

* **MakerHawk** – utilizado para a reprodução de um som de alarme de modo de aviso de perigo na área.
* **Mecanismo de fecho** – objetivo de controlo remoto/automático de fechar a porta caso exista perigo.
* **PIR HC-SE501** – para controlo de pessoas dentro da área de possível perigo, com o maior objetivo de impedir a porta se fechar se pessoas estiverem dentro da área.
* **Câmara RB-Camera\_JT** – utilizado para monitorização de uma ou várias válvula e retirar o valor dessas para avaliação de perigo da área.

**Testes executados:**

Neste momento fizemos testes à câmara que usamos para captar o valor da válvula, obtido pelo algoritmo com base no *Hough Transform* e notamos que a imagem presente no Raspberry Pi era decente mas quando a transmitirmos fora do Raspberry Pi a sua resolução e *Frames Per Second* (FPS) descobrimos que seria impossível a sua utilização, uma maneira de resolver o problema foi a transmissão, não de video continuo, mas de imagens retiradas dentro de pequenos intervalos de tempo de modo a conseguir receber com eficiência a imagem enviada e ao mesmo tempo manter a sua fidelidade do estado real.

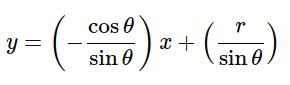
Outro teste realizado foi a ligação bluetooth a dispositivos que demos permissões de entrada especializados de modo a conseguirmos utilizar as funcionalidades de modo remoto, tivemos alguns problemas com ligação à Ethernet, dependendo se era da Instituto do ISPGaya ou alguns telemóveis/portáteis para conectar mas conseguimos resolver.

**Funcionalidades novas:**

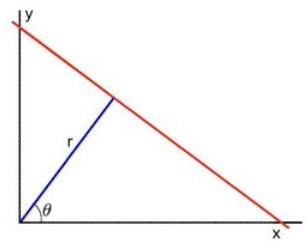
Neste momento não temos muito “novo” em relação ao relatório anterior devido ao foco de melhoramento e garantir que o que temos está em condições funcionais, mas temos a fonte externa de informação, a aplicação em flutter, que anteriormente está em fase de mockup e neste momento está desenhada. Outras funcionalidades é o começo da perspectiva de montagem em Breadboard os sensores de “alarme” e movimento (MakerHawk e o PIR HS-SE501 respectivamente), justamente com o código para ser testado quando os sensores estiverem presentes.

**Algoritmo usado:**

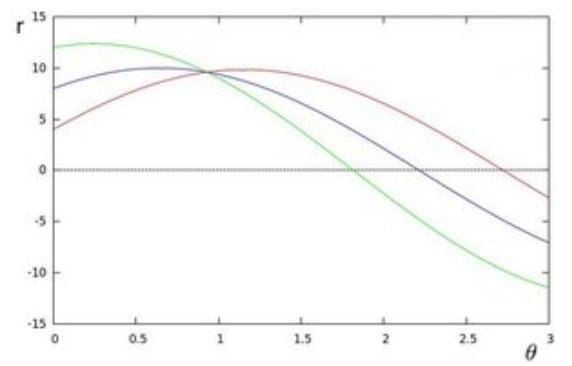
Para a detecção de válvulas, são usadas transformações *Hough*, que nos permitem extrair certos “*features*” da imagem, neste caso interessa-nos a face circular da válvula para posteriormente obtermos uma leitura da agulha. Apesar de estarmos a usar um framework de visão por computador(OpenCV), que nos abstrai das computações, é importante perceber exatamente como, e porque as transformações *Hough* funcionam.

Um exemplo simples da utilização de transformacoes de *Hough*, seria para a detecção de uma reta com a equação **y = mx + b**, em que no espaço de coordenadas cartesiano temos (**m**, **b**) como parâmetros de entrada. Para o efeito de transformações *Hough* iremos considerar a reta no sistema de coordenadas polar:

Simplificando a equação acima temos:

Onde **r** e **θ** sao os parâmetros de entrada e representam a distância da origem até a reta, e o ângulo com respeito ao eixo x respectivamente.

Se considerarmos um ponto (x0, y0), podemos definir o conjuntos de retas que passam no ponto (x0, y0) a partir da seguinte equação:

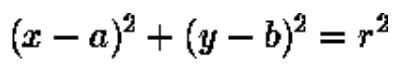
Onde cada par (**rθ, θ**) representa uma reta que passa no ponto (x0, y0).

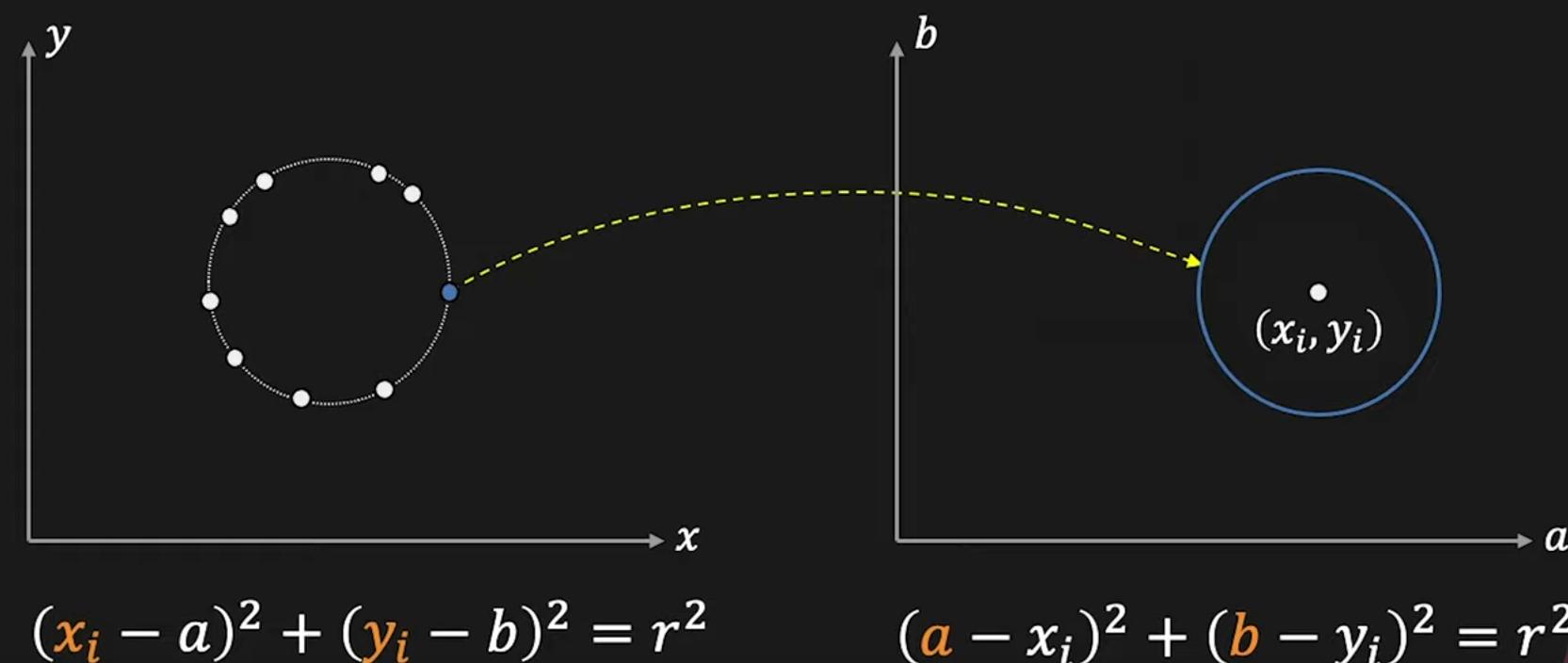
Olhando para o plot acima, para os pontos (x0 = 8, y0 = 6), (x1 = 4, y1 = 9), (x2 = 12, y2 = 3), podemos verificar que existe uma intersecção dos 3 plots no ponto x = 0.925, y = 9.6

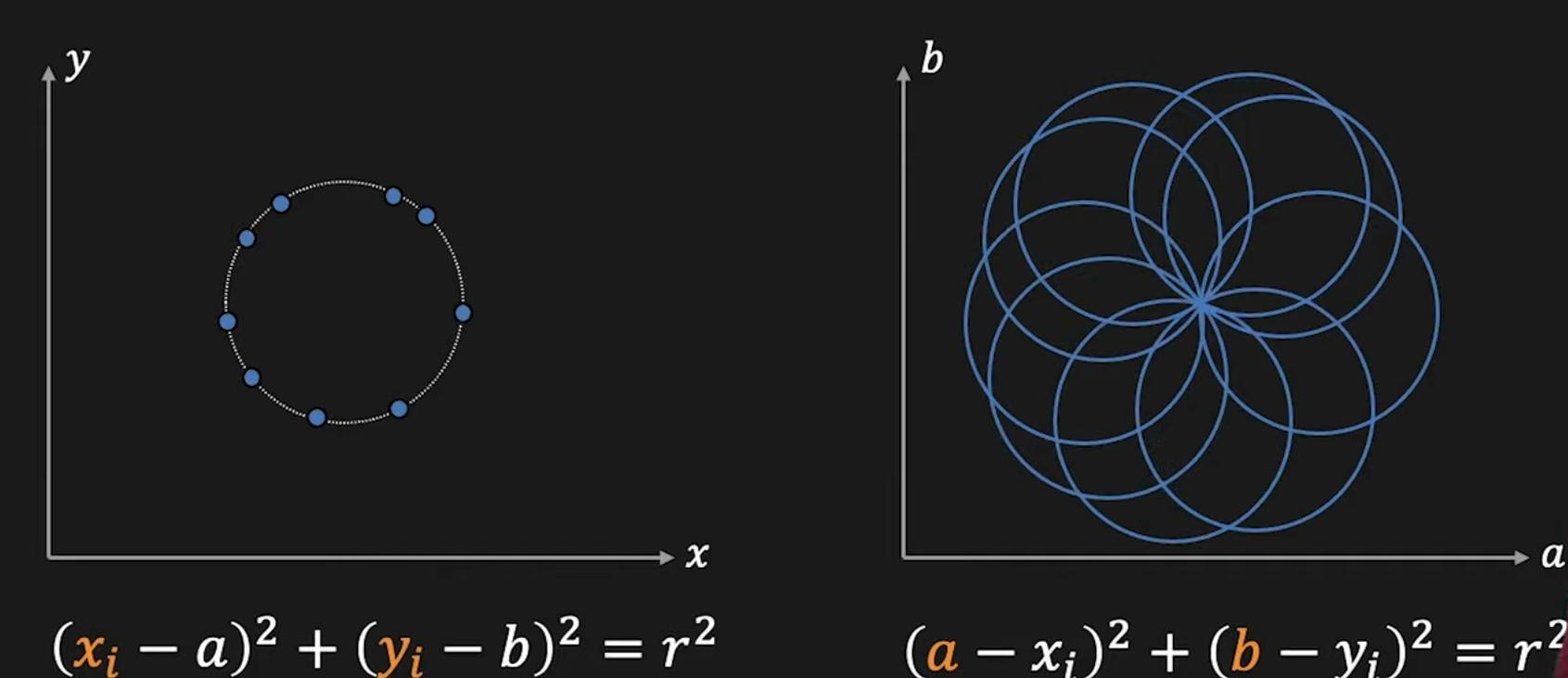
Analisando o plot acima, se existir um mínimo número de intersecções pode-se concluir que existe uma linha, com:

**r > 0, 0 < *θ* < 2*π***

Isto é o que o algoritmo de transformações *Hough* faz, mantém um registo de interseções entre curvas de cada ponto da imagem. Se esse número for superior a um certo limite, então conclui-se que temos uma reta com os parâmetros (**rθ, θ**).

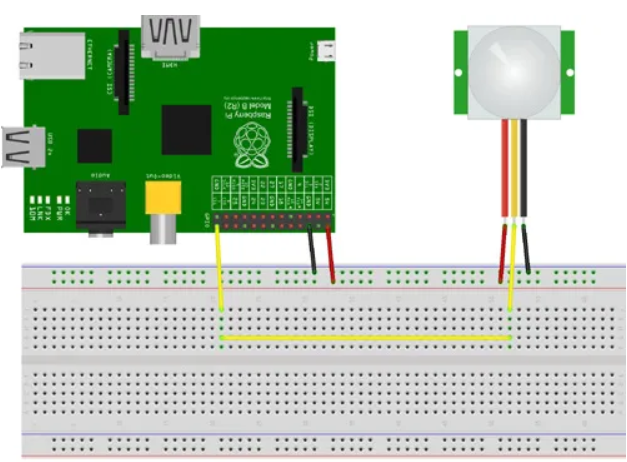
O mesmo princípio aplica-se para extrair features como círculos, para isso usamos a equação paramétrica do círculo:

****

****

**Esquemas:**

Em termos de esquemas o que temos para apresentar é mais teórico do que prático devido à obtenção dos sensores necessários para testes, mas podemos demonstrar a ideia que se irá desenvolver.



Aqui temos o sensor de movimento em que iremos ligar a partir da Breadboard de modo a conseguirmos conectar ao nosso Raspberry Pi.

Outro sensor é o de “alarme” que também iremos usar a Breadboard para ligarmos ao Raspberry Pi.

Diagrama

Descrição gerada automaticamente com confiança média

**Código relevante desenvolvido:**

Para facilitar o processamento do código com diversos membro foi criado um repositório na plataforma GitHub onde está disponibilizado o código utilizado para a aplicação em flutter e o Raspberry Pi:

<https://github.com/PauloBDev/gpr>

**Pesquisa de sensores / software:**

* **MakerHawk** – Necessidade de obtenção.
* **Mecanismo de fecho** – Necessidade de obtenção.
* **PIR HC-SE501** – Necessidade de obtenção.
* **Camara RB-Camera\_JT** – Obtido.

**Putty** – Software necessário para a utilização remota do RB\_Pi

**Flutter.dev –** Software Opensource para criação da aplicação.