Cabine Fotográfica Interativa

Jéssica Maciel,12/0014106 Mônica Damasceno, 10/0037097

***Resumo*—O objetivo deste projeto é construir uma cabine fotográfica, para ser usada em eventos, cujo produto final é gerar cinco fotos e um GIF. A câmera poderá ser acionada por sete *QR Codes* diferentes e cada um trará um efeito diferente para ser aplicado nas fotos. O controlador usado será a *Raspberry Pi* 3 e os arquivos obtidos serão armazenados na própria *Raspberry* e em uma conta de armazenamento em nuvem no *Dropbox* que os usuários terão acesso.**

Keywords*—*Raspberry Pi 3, fotos, cabine fotográfica, QR Code.

I. Introdução

O uso de cabine fotográficas tem se tornado cada vez mais popular em eventos, como casamentos e aniversários de 15 anos porque ela permite um ambiente personalizado e de acordo com o tema da festa.

A *Rapsberry* 3 é uma placa de dimensões pequenas e com grande capacidade de processamento (1.2GHz 64-bit *quad-core* ARMv8 CPU, 1GB RAM) [1] que permite embarcar diversas soluções. Com ela é possível modernizar as cabines fotográficas, usando-a para controlar o disparo das fotos da cabine aplicar filtros nas imagens, manipulá-las montando gifs e salvá-las no dispositivo escolhido.

Por isso, o objetivo deste trabalho é desenvolver um sistema de fotos automatizadas que permita ao usuário aplicar filtros à imagem, montar um gif com as imagens e salvar no seu dispositivo móvel para ser usado em cabine fotográfica de eventos.

Para tornar a cabina ainda mais interativa, o tipo de filtro aplicado nas imagens será escolhido por meio de um qrcode que estará disponível na formas das plaquinhas tão usadas nas festas.

Os requisitos do projeto são:

- Decodificar QRCode
- Aplicar filtro na imagem da câmera;
- Tirar fotos sequenciais;
- Montar um gif com as fotos registradas;
- Enviar as imagens e o gif para um sistema de nuvem;
- Disponibilizar o link para download para os usuários.

A cabine fotográfica interativa aqui apresentada é um produto portátil, que pode ser facilmente transportado de um lugar para outro e facilmente instalado, permite que sejam escolhidos previamente quais filtros serão disponibilizados aos usuários, automatiza o processo de disparo da câmera, do armazenamento das fotos em nuvem e torna as fotos mais acessíveis para que o usuário manipule, e baixe no seu computador e/ou celular instantes depois das fotos terem sido tiradas.

II. Desenvolvimento

A. Descrição do hardware

O *hardware* da cabine fotográfica é relativamente simples e seus principais componentes são a *Raspberry Pi* 3 e a câmera própria da placa.

Tabela I. Lista de materiais

Item	Descrição	Quantidade	Valor (R$)
1	Raspberry Pi 3	1	200,00
2	Módulo câmera para Raspberry Pi 5mp	1	45,00
3	Fonte de alimentação 5V | 3A	1	25,00
4	Cabo HDMI	1	30,00
5	Monitor LCD	1	160,00
6	Botão	1	1,00
7	Resistor de 1KΩ	1	0,05
8	Fios	1	3,00
9	Estrutura de madeira	1	20,00
		TOTAL	484,05

A tabela I apresenta os materiais utilizados no desenvolvimento da cabine.

Não foram necessários muitos componentes para desenvolver a cabine fotográfica, pois a câmera e a raspberry são responsáveis pela maior parte dos processos desenvolvidos no projeto.

O botão foi ligado a um pino digital 12 da *Raspberry Pi* 3 com um resistor *pull up* de 1KΩ ligada a placa por fios de cerca de 60 centímetros de comprimento, a câmera foi colocada no *slot* correspondente. A câmera foi fixada na parte frontal da estrutura de madeira da câmera e a raspberry, juntamente com os fios foi ocultada na parte de trás da estrutura de modo que apenas a câmera e o monitor ficassem visíveis para o usuário, conforme a figura 1. Além disso, o esquemático do projeto está representado na figura 2.

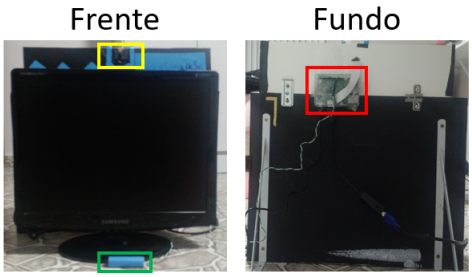

Figura 1. Projeto montado. Em amerelo, está destacado a câmera, em verde o botão e em vermelho a raspberry.

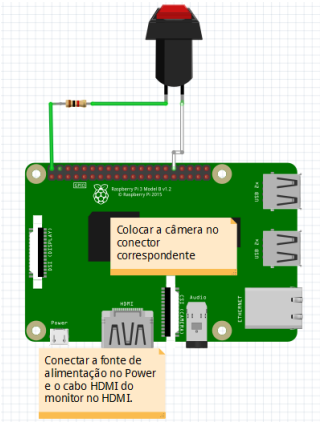

Figura 2. Esquemático do projeto, desenvolvido no software Fritzing

B. Descrição do software

A proposta deste projeto é que de acordo com a temática do evento em questão seja disponibilizado plaquinhas personalizadas, cada uma com um *QR Code* diferente. Os *QR Codes* levarão informações sobre o filtro a ser aplicado e após a leitura dará início aos disparos de 5 fotos sequenciais e criação de um gif. As fotos e gifs serão armazenadas na nuvem (*Dropbox*) sempre que houver internet. Para acionar o processo o usuário deverá apertar um botão.

Foram escolhidos nove diferentes filtros para aplicar nas fotos, um de ajuste de luz capturada na foto (*EV compensation*) que foi aplicado em todas as fotos, o efeito *antishake*, quatro de que aplicam diferentes efeitos nas fotos, sendo que um deles, o *blur*, foi aplicado em todas as fotos por fornecer uma foto mais limpa e bonita e os outros três (*colourswap, cartoon e o sketch*) tiveram sua aplicação condicionada aos *QR Codes* , dois de ajuste de temperatura (*sun e shade*), ambos condicionados à leitura do *QR Code* e os outros dois ("128:128" e -58:0) foram filtros de cor, o primeiro para deixar as fotos nas cores preto e branco e o segundo para aplicar um filtro azul na imagem.

A análise dos blocos - figura 3 e observações críticas serão descritas a seguir. Assim que se liga o sistema uma tela inicial com dicas e recomendações de uso é executada em paralelo com a leitura do botão com o uso de *threads*. Esta tela inicial é composta por um gif e chamada pela interrupção *System* e pelo comando *animate*.

O programa lê constantemente o sinal de um botão, caso seja apertado inicia-se o ciclo de leitura do *QR Code*. Caso contrário, a tela inicial continua sendo executada até ocorrer uma variação no sinal vindo do botão. Ou seja, se o botão é acionado durante a execução o programa espera o término do tela inicial para dar início a leitura.

O ciclo de leitura do *QR Code* consiste em tirar uma foto do usuário com o código. O programa *Zbar* avalia qual a informação do *QR Code* e a escreve em um arquivo de texto por meio da função *zbarimg*. Este arquivo de texto é lido e seu

Figura 3. Diagrama de blocos descrevendo o funcionamento do programa

conteúdo atribuído à uma variável para futuras comparações em laços condicionais.

Em posse da informação fornecida pelo qrcode se aplica os filtros e efeitos pertinentes por meio da função *if*. Foi adicionado dois laços de segurança para quando o valor lido pelo *QR Code* não for igual aos valores permitidos, uma mensagem de erro é retornada e o comando break é usado para que o *loop* recomece. Na tela aparecerá a imagem capturada pela câmera já com o filtro escolhido aplicado e as 5 fotos serão tiradas em um espaço de tempo determinado. As fotos são salvas e enviadas para uma pasta na *Raspberry*. Em seguida é criado um gif com as fotos presentes temporariamente nesta pasta. Enquanto o gif é gerado, ao usuário é apresentada uma tela de espera até que esteja pronto. O tempo de espera varia de acordo com o tamanho e qualidade da imagem.

Quando o gif com as fotos capturadas estiver pronto é exibido na tela por um número fixo de vezes. Paralelamente a *Raspberry* está fazendo uploading dos arquivos. Novamente são usadas *threads* para paralelizar as tarefas. A tarefa de exibição do gif produzido dura a quantidade de repetições que se estabeleceu baseado no tempo que leva o *uploading* de todos os arquivos.Neste projeto, optou-se por uma conta no *Dropbox* para armazenar as fotos e gifs na nuvem. O *upload* é realizado por um aplicativo que deve ser instalado e configurado na *Raspberry*. O *link* desta conta ficaria acessível para todos que quisessem fazer *download* das fotos e gifs.

Em sequência os arquivos gravados na *Raspberry* são transferidos para outras duas pastas, uma para as fotos e outra para os gifs, dentro desse mesmo diretório.

Cumpre-se assim todos os requisitos propostos, com algumas observações para futuros aprimoramentos. Por exemplo, a otimização do tempo de espera para criação do gif e uma maneira de evitar que se alterne a tela principal com o terminal ao final de alguns processos.

III. Resultados

O primeiro teste realizado foi tirar fotos sequenciadas com a câmera controlada por um botão. Após validar os comandos para realizar tal operação, foi-se sem em busca de um programa que montasse o gif das imagens.. Pela facilidade de uso, o *graphicsmagick* foi escolhido.

O passo seguinte foi a leitura do *QR Code* e para isso, os fóruns pesquisados indicaram que o *zbar* era o programa

mais indicado. Porém, a instalação do programa realizou-se de forma bastante complexa. Para o correto funcionamento do programa era necessário que ele criasse um arquivo (video0) na pasta /dev, no entanto, após diversas tentativas o arquivo não era criado.

A tentativa seguinte foi de instalar o módulo "bcm2835-v4l2" que eventualmente permitia a criação do arquivo e o consequente funcionamento correto do *ZBar*. No entanto, o recurso nem sempre funcionava e a solução não poderia ser utilizada.

Após esgotadas as possibilidades, o sistema operacional da *Raspberry* - *Raspbian* - foi formatado. Após a formatação, o aplicativo *ZBar* foi reinstalado e funcionou corretamente.

No entanto, a lógica do programa estava passando o controle da câmera para que o aplicativo identificasse o *QR Code*. Essa lógica não funcionou muito bem porque ao ser chamada pelo *system* ele iniciava a câmera e mesmo após a leitura e identificação do código ele não saia do controle da câmera e impedia a continuação do programa.

A fim de evitar tal problema, a identificação do código passou a ser realizado por meio de uma imagem enviada para o *zbar*. Assim, o programa passou a realizar a leitura do código sem ter o controle da câmera. Os resultados com tal abordagem foram muito bem sucedidos tanto que essa foi a lógico utilizada no código final.

Por fim, a última parte que restava para a execução correta da cabine era o *upload* dos arquivos uma nuvem. Como o programa foi todo desenvolvido na linguagem C, buscou-se um servidor que tivesse uma suporte à comunicação em C, nessa busca, foi encontrado o *Dropbox*. Com os testes de comunicação bem sucedidos, criou-se uma conta no servidor específica para a cabine fotográfica e estabeleceu-se a comunicação.

Após unir as quatro partes - ler botão, ler *QR Code*, tirar fotos, montar gif e subir as fotos, percebeu-se que em dois momentos, montagem do gif e *upload* dos arquivos, havia um período muito grande com a tela do terminal. Por isso, foram montados dois gifs para rodar em paralelo a esses momentos de modo o programa se torne mais amigável para o usuário.

Com isso, a versão final do programa, disponível no ANEXO I, cumpriu seu objetivo final, apesar de ainda apresentar um latência considerável durante a leitura do botão, montagem do gif e *upload* dos arquivos.

A integração dos componentes eletrônicos na estrutura da cabine permitiu que o projeto se tornasse mais móvel, podendo ser instalado em qualquer local próximo de uma tomada. Além disso, após o *upload* as fotos rapidamente ficaram disponíveis em qualquer dispositivo móvel com *internet* no *link* compartilhado da pasta do *Dropbox*.

Além disso, a partir da lista de materiais descrita na tabela I, percebe-se que foi desenvolvido a parte central da cabina, sem a parte estrutural típica de uma cabine fotográfica, como as cortinas para torná-la mais privada, os bancos para as pessoas tirarem fotos sentadas e sem os adereços tão procurados nesse tipo de fotos. Se considerarmos que esses itens custam cerca de 500 reais e que o aluguel de uma cabine fotográfica custa cerca de 1200 reais para quatro horas de festa, percebe-se que em apenas um evento o produto já retorna o valor gasto na sua construção física, sem incluir os gastos com o desenvolvimento de *software*. Demonstrando que a cabine fotográfica interativa é um produto viável.

IV. Conclusão

O projeto de desenvolvimento de uma cabine fotográfica na *Raspberry* possui algumas facilidades, uma vez que a própria câmera já aplica alguns filtros durante a captura das imagens. No entanto, a integração desses recursos com outros recursos, como o da leitura do *QR Code* e o *upload* pode ser um pouco mais trabalhosa.

Além disso, o tempo de execução de alguns programas é um fator um pouco incômodo que pode ser ajustado após uma análise mais profunda dos códigos a fim de se realizar uma otimização dos mesmos.

Apesar disso, o desenvolvimento da cabine fotográfico envolveu um processo de muito aprendizado como todos os erros que surgiram e foram solucionados e, após o funcionamento correto, os testes com pessoas sem conhecimento de programação que se voluntariaram para testar a cabina a fim de apontar melhorias, mostrou que uma boa aceitação e rendeu momentos de muita descontração, atingindo o objetivo da cabine que é a de servir como instrumento de descontração para os usuários.

Referências

[1] E. Upton, G. Halfacree, "Raspberry PI Manual do Usuário", São Paulo, 1 ed, 2013.