Universidade Federal do ABC
Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas
Trabalho de graduação em Engenharia da Informação

Medidor de glicose compacto com comunicação serial OTG

Muriel Da Costa Santos

Santo André - SP
2023

Muriel Da Costa Santos

Medidor de glicose compacto com comunicação serial OTG

Trabalho de graduação apresentado ao curso de Engenharia de Informação da Universidade Federal do ABC, como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia de Informação.

Universidade Federal do ABC – UFABC

Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas

Programa de Graduação em Engenharia da Informação

Orientador: Marcelo Bender Perotoni

Santo André - SP

2023

Muriel Da Costa Santos

Medidor de glicose compacto com comunicação serial OTG

Trabalho de graduação apresentado ao curso de Engenharia de Informação da Universidade Federal do ABC, como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia de Informação.

Trabalho aprovado. Santo André - SP, 24 de dezembro de 2023:

Marcelo Bender Perotoni
Orientador

Professor
Cláudio José Bordin Jr

Professor
Marco Cazarotto

Santo André - SP
2023

Agradecimentos

Agradeço primeiramente a Deus.

Agradeço aos meus pais e minha irmã, pelo apoio incondicional, que me ensinaram a amar e mudar as coisas.

Agradeço ao meu orientador, Marcelo Bender Perotoni, por todos os conselhos, pela paciência e ajuda nesse período.

Aos meus amigos de curso, em especial Lucas Mieri que me motivou nos momentos difíceis.

Aos professores do curso de Engenharia de Informação por todo conhecimento compartilhado.

Resumo

Este trabalho explora o desenvolvimento de um dispositivo medidor de nível de glicose no sangue, a fim de minimizar o custo de acesso do equipamento para pessoas diabéticas que realizam o controle diário da glicose. Também visa diminuir o tamanho do equipamento com o intuito de prover maior mobilidade aos usuários e, consequentemente, aumentar a frequência no uso do aparelho. A premissa é de que o dispositivo opere em conjunto com um *smartphone*, para onde será enviado resultados de medição através de comunicação serial, tornando os dados mais acessíveis de serem compartilhados com eventuais sistemas de terceiros.

Palavras-chaves: glicosímetro. OTG. smartphone. compacto.

Abstract

This work explores the development of a blood glucose level measuring device, in order to minimize the equipment cost for diabetic patients, who need it for daily control. It also aims to reduce the size of the equipment in order to provide greater mobility to users and consequently increase the frequency of use of the device. The premise is that the device operates in conjunction with a smartphone, where measurement results will be sent via serial communication, making the data more accessible to be shared with any third-party systems.

Keywords: glucometer. OTG. glucose meter.

Lista de ilustrações

Lista de abreviaturas e siglas

USB	***Universal Serial Bus*** - Barramento serial de comunicação que permite a conexão e comunicação entre dispositivos eletrônicos. É composto por um conjunto de pinos que podem ser usados para transferência de dados e fornecimento de energia elétrica.
OTG	***On The Go*** - Tecnologia que permite a conexão de dispositivos eletrônicos móveis sem a necessidade de um computador intermediário.
TTL	***Transistor-Transistor Logic*** - Circuito eletrônico digital que é baseado em transistores bipolares. É comumente utilizado em projetos de circuitos lógicos.
RS-232	Padrão de comunicação serial usado para transmitir dados entre dispositivos eletrônicos.
UART	***Universal Asynchronous Receiver Transmitter*** - Circuito eletrônico usado para implementar a comunicação serial assíncrona entre dispositivos. Possui um transmissor e um receptor que são usados para enviar e receber dados em série. Ele converte os dados paralelos em um formato serial e vice-versa.
CPU	***Central Processing Unit*** - Componente principal de um computador ou dispositivo eletrônico responsável por executar instruções e processar dados. É responsável por todas as operações lógicas e aritméticas que são executadas.
ADC	***Analog to Digital Converter*** - Circuito eletrônico que converte um sinal analógico em um sinal digital.
PWM	***Pulse-Width Modulation*** - Técnica eletrônica que consiste em variar a largura de um pulso de sinal em um intervalo de tempo fixo, de forma a modificar a quantidade de energia transmitida.
IDE	***Integrated Development Environment*** - Software que fornece um ambiente integrado para desenvolvimento de software.
CI	**Circuito Integrado** - Dispositivo eletrônico que contém vários componentes eletrônicos, tais como transistores, resistores e capacitores, interconectados por trilhas condutoras e encapsulados em uma única peça de silício.

AC	***Alternating Current*** - Tipo de corrente elétrica que muda de direção e magnitude em ciclos regulares.
DC	***Direct Current*** - Tipo de corrente elétrica que flui em uma única direção constante, sem mudanças de polaridade.
GND	***Ground*** - Ponto de referência de potencial elétrico comum, utilizado como uma referência de zero volts para todas as tensões de um circuito.
SMD	***Surface mount Device*** - Componente eletrônico projetado para ser soldado diretamente na superfície de uma placa de circuito impresso, em vez de ser inserido através de furos.
PCB	***Printed Circuit Board*** - Placa isolante que possui trilhas condutoras de cobre que permitem a conexão elétrica entre os componentes eletrônicos presentes em um circuito.
EEPROM	***Electrically Erasable Programmable Read Only Memory*** - Tipo de memória não volátil que pode ser apagada e reprogramada eletronicamente.
TIA	***Transimpedance amplifier*** - Amplificadores que convertem corrente em tensão.
LED	***Light Emitting Diode*** - Diodo emissor de luz.
GBW	***Gain bandwidth product*** - produto ganho-largura de banda.
DDP	**Diferença de potencial** - Diferença de potencial elétrico.

Sumário

1 Introdução

1.1 Quadro clínico

A diabetes é uma doença cujo principal indicativo é o aumento da glicose no sangue - hiperglicemia. O pâncreas é o órgão responsável por produzir o hormônio da insulina no corpo humano através das células beta e a insulina é responsável pela manutenção dos níveis de glicose no sangue, estimulando as células do corpo a absorverem a glicose (BRASIL, 2002).

De acordo com o Atlas do Diabetes da Federação Internacional de Diabetes, em 2021 havia 15,7 milhões de adultos vivendo com a doença no Brasil, onde ocupa a quinta posição no ranking dos países com maior incidência de diabetes no mundo. A estimativa é que a incidência da doença em 2045 chegue a 23,2 milhões (IDF, 2021) conforme figura 1.

Figura 1 – Os 10 principais países ou territórios em número de adultos (20–79 anos) com diabetes em 2021 e 2045.

2021			2045		
Rank	**Country or territory**	**Number of people with diabetes (millions)**	**Rank**	**Country or territory**	**Number of people with diabetes (millions)**
1	China	140.9	1	China	174.4
2	India	74.2	2	India	124.9
3	Pakistan	33.0	3	Pakistan	62.2
4	United States of America	32.2	4	United States of America	36.3
5	Indonesia	19.5	5	Indonesia	28.6
6	Brazil	15.7	6	Brazil	23.2
7	Mexico	14.1	7	Bangladesh	22.3
8	Bangladesh	13.1	8	Mexico	21.2
9	Japan	11.0	9	Egypt	20.0
10	Egypt	10.9	10	Turkey	13.4

Fonte: (IDF, 2021)

Existem 4 tipos de diabetes, porém os mais comuns são a diabetes tipo 1 e 2. A diabete tipo 1 é caracterizada pela baixa produção de insulina, causado por anticorpos produzidos pelo próprio corpo humano e que atacam as células beta do pâncreas, reduzindo assim a sua capacidade de produção de insulina (BRASIL, 2002).

Na diabete tipo 2, por sua vez, os pacientes não têm dificuldade na produção do hormônio da insulina, porém o organismo do indivíduo possui resistência à mesma. Nesse caso o pâncreas eleva cada vez mais a produção. Quando o órgão não possui mais

capacidade de suprir os níveis elevados de insulina para manter estável a glicose, surge a diabetes (BRASIL, 2002).

A glicose é a principal fonte de energia do corpo e monitorar seus níveis regularmente no sangue pode ajudar as pessoas com diabetes a evitar problemas de saúde e ajustar sua dieta, medicação e estilo de vida para manter seus níveis de açúcar no sangue sob controle. Se os níveis de glicose no sangue não forem mantidos dentro da faixa normal, isso pode acarretar em problemas de saúde a curto e longo prazo, como:

- Hipoglicemia: níveis muito baixos de açúcar no sangue, o que pode levar a sintomas como tonturas, fadiga, suor frio e, em casos graves, convulsões ou coma.
- Hiperglicemia: níveis muito elevados de açúcar no sangue, o que pode levar a sintomas como sede excessiva, micção frequente, visão embaçada e, em casos graves, cetoacidose diabética, um estado de emergência médica que pode levar ao coma e até mesmo à morte.
- Danos nos órgãos: níveis elevados de açúcar no sangue podem danificar os vasos sanguíneos e órgãos do corpo a longo prazo, incluindo os olhos, rins, nervos e coração.

Tendo em vista a necessidade do controle dos níveis de glicose no sangue, este trabalho visa o desenvolvimento de um medidor de glicose de baixo custo, com uso de tecnologias modernas para miniaturização e visualização dos dados.

1.2 Motivação

Este projeto tem como premissa diminuir o tamanho do aparelho medidor de glicose e salvar os dados individuais de cada usuário em sua jornada diária de acompanhamento da medição de glicose, de forma que possa ser compartilhado com bancos de dados a fim de torná-los mais úteis para auxiliar em pesquisas e no desenvolvimento científico de modo geral. Um banco de dados com resultados de medição de glicose que possa ser utilizado por pesquisadores e empresas seria útil em várias áreas, especialmente para gerenciar e melhorar o controle da diabetes. Algumas das possíveis utilidades são:

- Monitoramento do controle glicêmico: Os resultados de medição de glicose podem ser armazenados no banco de dados para acompanhar as flutuações dos níveis de glicose no sangue ao longo do tempo. Isso pode ajudar a detectar padrões, identificar variações e entender como diferentes fatores, como alimentação, exercícios e medicamentos, afetam os níveis de glicose e o tratamento da diabete.
- Facilitar a tomada de decisões: Um banco de dados com informações sobre os níveis de glicose pode ser usado para gerar relatórios e gráficos que podem ser compartilhados

com o médico, o que facilita a tomada de decisões relacionadas ao tratamento. Isso permite que o profissional da saúde ajuste o tratamento de acordo com a resposta individual de cada paciente, com base nos dados, o que pode ajudar a melhorar o controle da diabetes e outras doenças que fazem o monitoramento da glicose.

- Identificação de tendências: Um banco de dados pode ajudar a identificar tendências em relação à saúde do paciente e a prever os riscos associados à diabetes, como complicações cardiovasculares, neuropatias, retinopatias, entre outras.

- Pesquisa científica: Um banco de dados com informações sobre os níveis de glicose pode ser usado por pesquisadores para estudar o comportamento da diabetes em diferentes populações, para desenvolver novas terapias, tratamentos e soluções inovadoras para a saúde humana. Em resumo, um banco de dados com resultados de medição de glicose pode ajudar no gerenciamento da diabetes e melhorar o controle glicêmico, permitindo uma melhor comunicação entre o médico e o paciente, além disso também pode possibilitar uma melhor compreensão da doença, permitindo avanços tecnológicos e científicos.

Na figura 2 podemos ver um exemplo comercial de kit de medição de glicose composto por lancetas, lancetador, tiras de medição, manual de instruções e estojo para transporte.

Figura 2 – Kit medidor de glicose comercial.

Fonte: (ACCUMED, 2023)

O kit do aparelho da figura 2 é volumoso a ponto não ser possível colocar no bolso. Um aparelho de glicose menor e portátil que possa ser colocado no bolso pode trazer várias vantagens, tais como:

- Mais discreto: Um aparelho de glicose menor seria menos perceptível e menos invasivo do que os aparelhos maiores e mais volumosos, o que poderia reduzir o estigma associado ao uso de aparelhos de glicose e permitir que as pessoas se sintam mais confortáveis em usá-los em público.
- Facilidade de uso: Um aparelho de glicose menor e portátil poderia ser mais fácil de usar do que os modelos maiores, pois seria mais simples de manusear e poderia ter recursos adicionais, como um visor maior e mais claro, que tornariam a leitura dos resultados mais fácil.
- Monitoramento mais frequente: Um aparelho de glicose menor e portátil pode encorajar as pessoas a monitorarem seus níveis de glicose com mais frequência, o que pode levar a um melhor controle da diabetes e prevenção de complicações de saúde a longo prazo. Além disso, a medida mais frequente permite ao usuário identificar alimentos e hábitos pessoais que impactem negativamente no controle da diabetes.
- Menor custo: Um aparelho de glicose menor e portátil pode ser menos dispendioso do que os modelos maiores, o que pode torná-lo mais acessível para pessoas que lutam para pagar por suprimentos e equipamentos médicos.

2 Fundamentação teórica

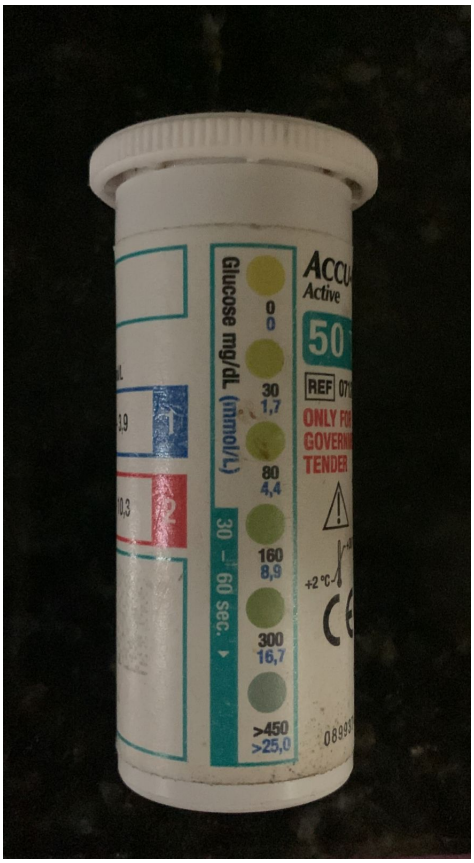
O medidor de glicose ou glicosímetro, é um dispositivo médico projetado para determinar a concentração de glicose no sangue de uma pessoa. Isto é feito coletando uma pequena amostra de sangue, geralmente obtida através de um furo no dedo, e aplicando essa amostra em uma tira reagente. O medidor então realiza uma análise química ou eletroquímica dessa amostra para quantificar a quantidade de glicose presente no sangue. O resultado é exibido em uma tela digital como um valor numérico que representa a concentração de glicose em miligramas por decilitro (mg/dL) ou milimoles por litro (mmol/L), dependendo da unidade de medida utilizada no país.

2.0.1 Princípios de funcionamento do glicosímetro

Os glicosímetros utilizam diferentes princípios de funcionamento para medir os níveis de glicose no sangue. Dois desses princípios comuns em equipamentos portáteis são o amperométrico e o colorimétrico.

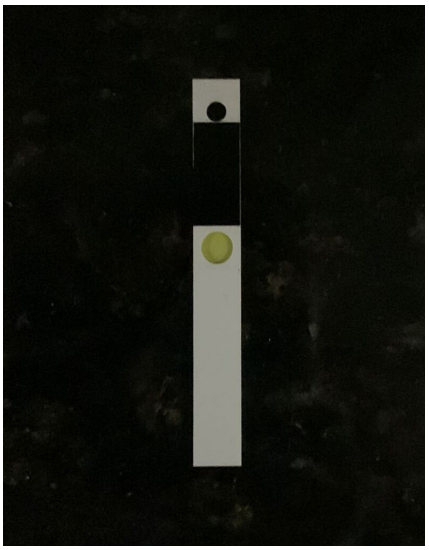
No método colorimétrico, a tira reagente contém substâncias químicas que reagem com a glicose na amostra de sangue, resultando em uma alteração de cor na substancia presente, conforme tira da figura 4. O princípio da reflexão de cor é empregado neste método para avaliar a intensidade cromática na camada de reação da tira de teste por meio da fotometria, onde um fotodetector no glicosímetro avalia a intensidade e a escala da cor presente e a compara com uma escala pré-calibrada para determinar a concentração de glicose (MICROCHIP, 2013). Na figura 3 é possível ver esta escala em uma embalagem de tiras reagentes que utiliza o método colorimétrico.

Figura 3 – Embalagem com escala de glicose de tiras reagentes do método colorimétrico.



Fonte: Elaborado pelo autor (2023)

Figura 4 – Tira reagente de glicose do método colorimétrico.



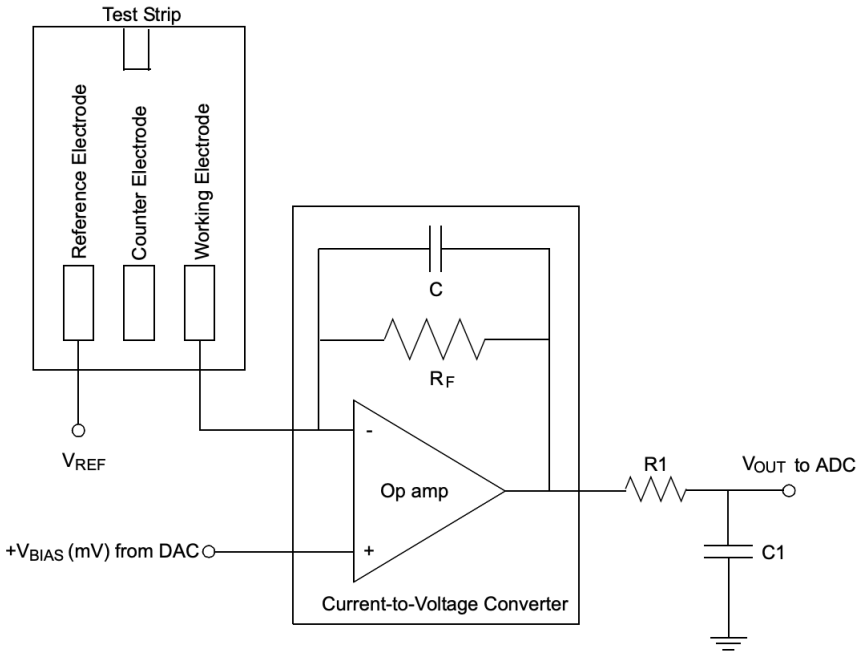
Fonte: Elaborado pelo autor (2023)

No método amperiométrico, a tira de teste eletroquímica incorpora um canal capilar destinado a absorver a solução aplicada em uma das extremidades da tira (MICROCHIP, 2013). A tira de medição de glicose tem duas camadas condutoras de eletricidade: uma camada superior e uma camada inferior. A camada superior é revestida com uma solução que contém uma enzima chamada glicose oxidase. Quando a gota de sangue é aplicada à camada superior, a glicose na amostra reage com a enzima e é convertida em ácido glicurônico e peróxido de hidrogênio (WILSON; TURNER, 1992).

O peróxido de hidrogênio gerado na reação então difunde-se pela tira e atinge a camada inferior. A camada inferior é revestida com uma solução contendo uma enzima chamada peroxidase e um agente redutor. A enzima peroxidase reage com o peróxido de hidrogênio para oxidar o agente redutor e, assim, produzir elétrons livres (BANKAR MAHESH V. BULE; ANANTHANARAYAN, 2009).

A corrente elétrica gerada pela presença de elétrons livres é proporcional à quantidade de glicose na amostra de sangue, e é mensurada através de uma interface de transimpedância que converte corrente em tensão elétrica, conforme podemos ver na figura 5. A leitura da corrente elétrica é convertida em um valor numérico correspondente ao nível de glicose no sangue e exibido na tela do medidor.

Figura 5 – Esquema elétrico de medição da tira de teste.



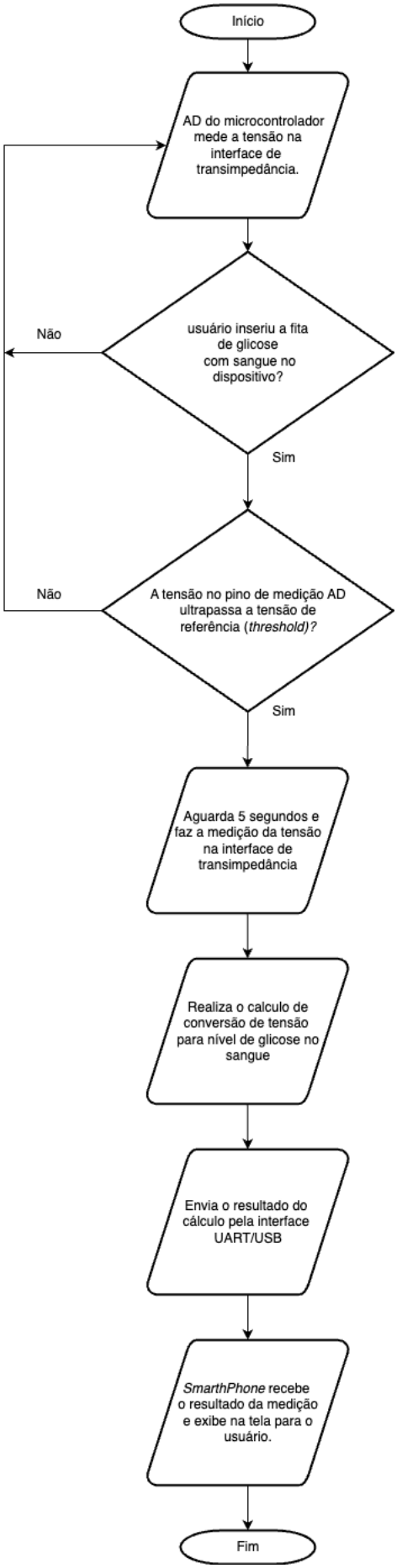
Fonte: (MICROCHIP, 2013)

Este trabalho utiliza a abordagem amperiométrica, porém ambos os métodos têm sido amplamente utilizados em medidores de glicose ao longo dos anos, com o objetivo comum de fornecer medições precisas e rápidas dos níveis de glicose no sangue.

3 Desenvolvimento

O dispositivo proposto possui o funcionamento detalhado conforme o diagrama em blocos da figura 6. O usuário deve inserir a amostra de sangue na tira de teste e, o sistema irá processar e enviar os dados resultantes para o celular, onde será possível visualizar na tela o valor de glicose no sangue.

Figura 6 – Diagrama em blocos do sistema proposto.



Fonte: Elaborado pelo autor (2023)

3.1 Comunicação entre aplicativo e medidor de glicose

A comunicação utilizada será a OTG, uma tecnologia que permite a conexão de dispositivos eletrônicos móveis, como *smartphones* e *tablets*, a outros dispositivos, como câmeras digitais, teclados e *drivers* USB, sem a necessidade de um computador intermediário.

Com a comunicação OTG, o dispositivo móvel pode atuar como um *host* ou dispositivo periférico, dependendo do tipo de dispositivo com o qual está se comunicando. Por exemplo, se um *smartphone* estiver conectado a uma câmera digital usando um cabo OTG, o *smartphone* atua como *host* e a câmera digital como periférico, permitindo que o *smartphone* controle a câmera e acesse suas fotos e vídeos.

A comunicação OTG é especialmente útil para usuários que precisam transferir arquivos ou dados entre dispositivos móveis sem a necessidade de um computador intermediário. Também serve para conectar dispositivos de entrada, como teclados e mouses, a dispositivos móveis para uma melhor experiência de digitação ou controle.

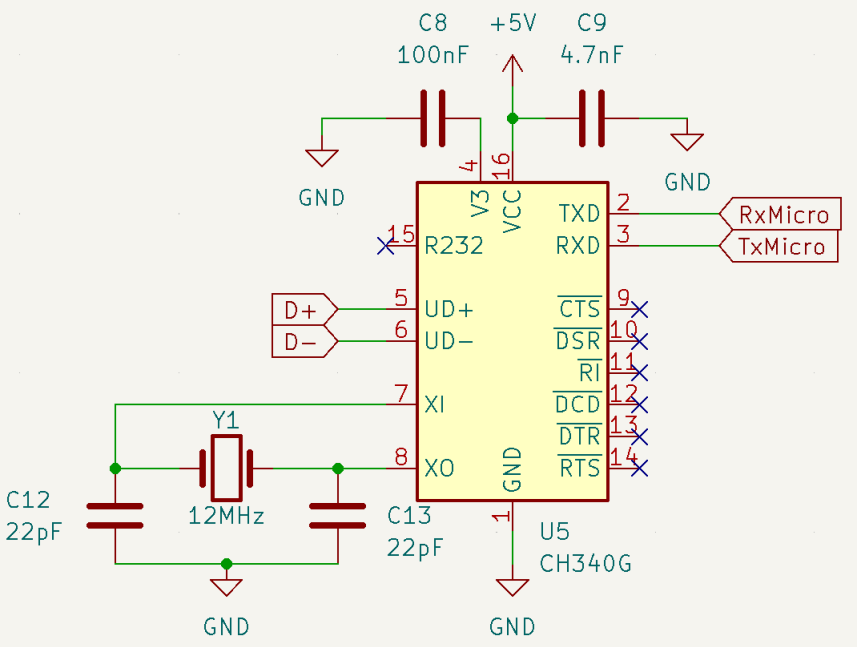
O dispositivo medidor de glicose possui um microcontrolador que envia informações utilizando comunicação serial UART, uma interface de comunicação que permite a transferência de dados em série bit a bit, entre dispositivos eletrônicos. Os *smartphones* modernos geralmente não possuem uma entrada serial tradicional, como as portas seriais RS-232 que eram comuns em computadores mais antigos. No entanto, muitos *smartphones* têm portas USB que suportam a comunicação serial USB, o qual é uma interface de comunicação em série usada para conectar dispositivos eletrônicos a um computador ou outro dispositivo hospedeiro. Neste caso será utilizado o circuito integrado CH340G para fazer a interface de conversão entre o USB do *smartphone* e a UART do microcontrolador.

O CH340G converte os sinais USB em sinais de comunicação serial TTL e vice-versa, permitindo que um microcontrolador ou outro dispositivo eletrônico com uma porta serial TTL se comunique com um computador ou outro dispositivo USB. Ele é compatível com velocidades de transferência de dados de até 2 Mbps e suporta os protocolos de comunicação RS-232, RS-485 e RS-422. (DREAMCITY, 2014)

Uma das principais vantagens do CH340G é que ele é relativamente fácil de usar e pode ser programado para funcionar com uma variedade de dispositivos e aplicações diferentes. O CH340G também pode ter problemas de compatibilidade com alguns sistemas operacionais e requer a instalação de *drivers* específicos para funcionar corretamente, mas não foram encontrados problemas relacionados a compatibilidade com o *Android*, já que este sistema utiliza o *kernel* Linux, onde quase todos os *drivers* disponíveis são incorporados diretamente no *kernel*.

Na figura 7 é possível visualizar o circuito de comunicação projetado.

Figura 7 – Circuito conversor serial-USB utilizando CI CH340G.



Fonte: Elaborado pelo autor (2023)

Os pinos 2 e 3 estão conectados ao microcontrolador para trabalhar com a comunicação *full-duplex* UART, enquanto os pinos diferenciais 5 e 6 estão ligados diretamente aos pinos do conector que faz a ligação com o USB do celular. O capacitor C9 foi adicionado na configuração de desacoplamento como precaução para evitar ruídos contaminem a alimentação DC do CI. O cristal de 12MHz e os demais capacitores são requisitos mínimos especificados no *datasheet* do fabricante para o funcionamento adequado. (DREAMCITY, 2014)

3.2 Fonte de alimentação

Os pinos do conector micro USB fornecem uma alimentação para dispositivos que se conectam a eles. A alimentação de 5V é fornecida através dos pinos VCC (pino 1) e GND (pino 4 e 5 são curto circuitados) do conector USB, conforme figura. 8.

Figura 8 – Pinout do conector micro USB.

Pin	Name	Cable color	Description
1	VCC	Red	+5 VDC
2	D-	White	Data -
3	D+	Green	Data +
4	ID		Mode Detect
5	GND	Black	Ground

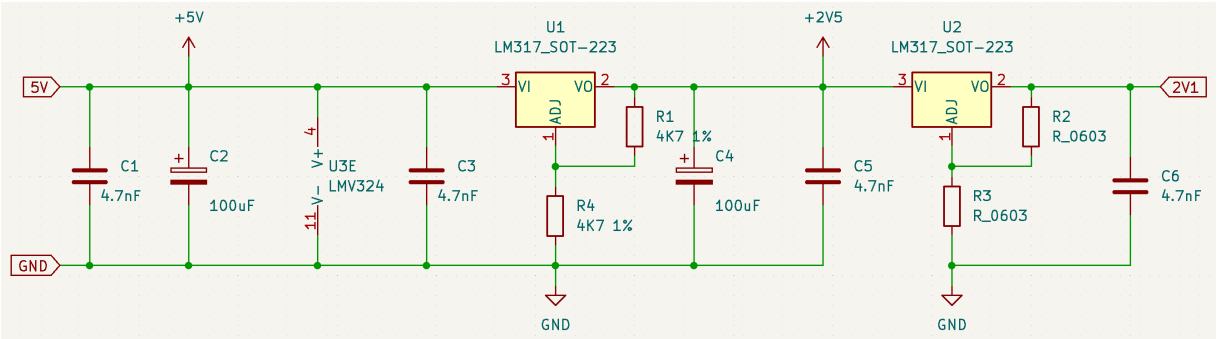
Mode Detect. May be N/C, GND or used as an attached device presence indicator (shorted to GND with resistor)

Fonte: (SUNROM, 2022)

É importante observar que a corrente máxima que pode ser fornecida pelos dispositivos OTG é de 500mA a 5V. No entanto, a maioria dos dispositivos modernos utilizam de técnicas de gerenciamento de energia para limitar o consumo de corrente a um nível mais baixo, a fim de evitar sobrecarga do circuito de alimentação USB. (TI, 2010)

Na figura 9 é possível visualizar o circuito da fonte de alimentação desenvolvida para este projeto.

Figura 9 – Fonte de alimentação com reguladores lineares.



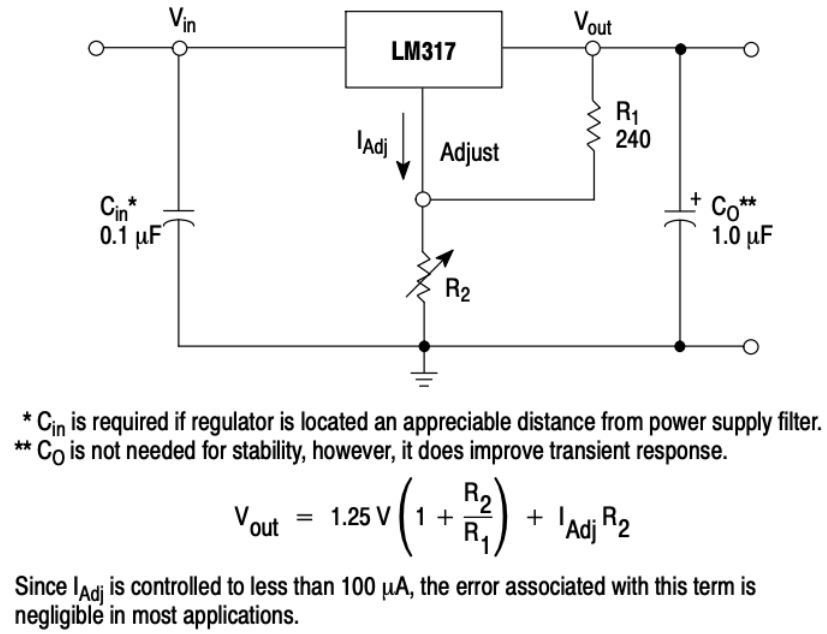
Fonte: Elaborado pelo autor (2023)

Neste projeto são necessárias mais duas tensões de 2,5V e 2,1V além dos 5V fornecidos pelo USB, que são utilizadas como referência para o condicionamento de sinal do transdutor de transimpeância. Para isto estão sendo utilizado os reguladores lineares

U1 e U2. O CI LM317 é capaz de fornecer uma tensão regulada com valores variando de 1,2V até 37V, com uma corrente máxima de 1,5A. O LM317 é um regulador de tensão linear, que significa que ele regula a tensão de saída ajustando a resistência elétrica entre a entrada e a saída do circuito, dissipando a diferença de potencial como calor. Para ajustar a tensão de saída do LM317, é utilizado um divisor de tensão cuja alimentação é a própria saída, servindo de tensão de *feedback*. (ONSEMI, 2021)

O cálculo utilizado para os resistores do divisor de tensão estão evidenciados na figura 10, conforme o *datasheet* do fabricante informa. O valor da corrente ADJ é abaixo de 100μA e para efeitos práticos foi desprezada no cálculo.

Figura 10 – Orientação do fabricante para cálculo do dos resistores de ajuste de tensão.



Fonte: (ONSEMI, 2021)

Os capacitores cerâmicos em paralelo com as linhas de alimentação estão dispostos como desacopladores, a fim de curto circuitar variações de tensão provenientes de ruído de alta freqûencia, ou seja, quando for detectado ruído harmônico os capacitores irão filtra-los. Os capacitores eletrolíticos são para evitar quedas de tensão que podem ser ocasionados tanto pela alimentação USB, quanto por picos de consumo do circuito do aparelho. (ANALOG, 2009)

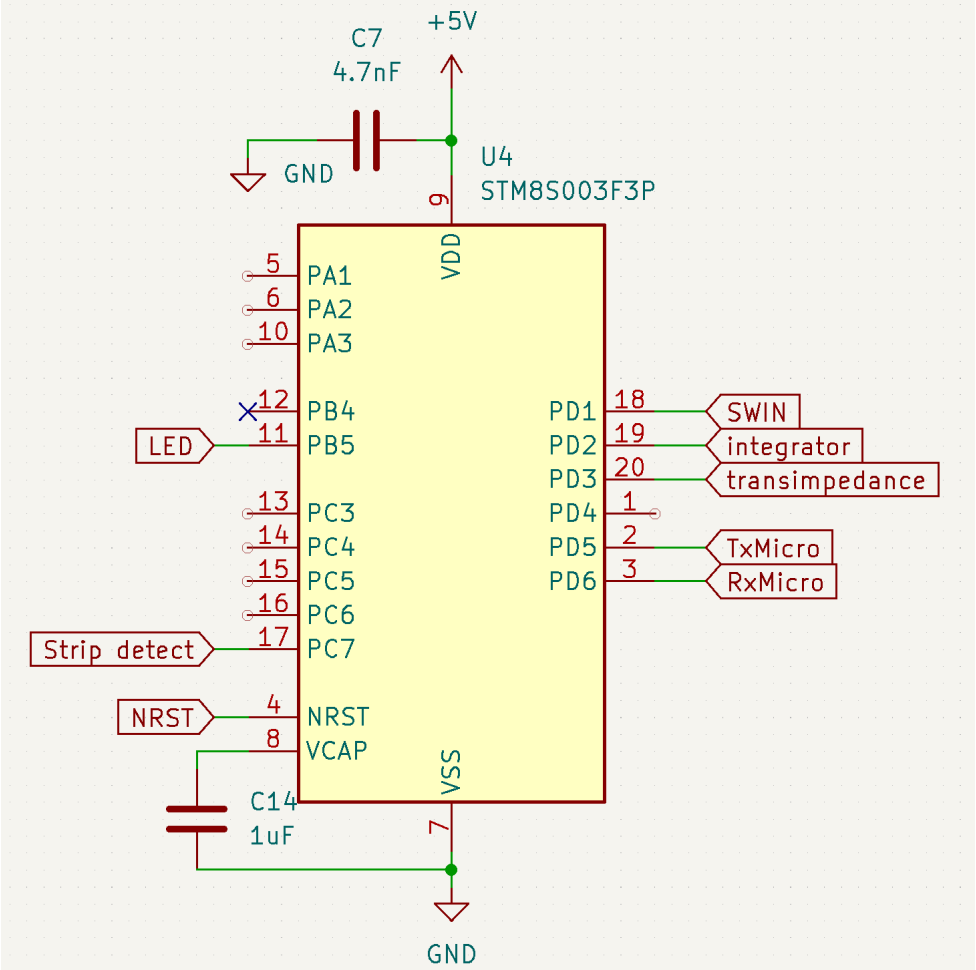
3.3 Processamento da informação

O microcontrolador escolhido foi o SMT8S, que é responsável por todo processamento computacional do projeto. O microcontrolador STM8S003F3P é um dispositivo da família STM8 da *STMicroelectronics*, projetado para aplicações de controle em sistemas embarcados de baixo custo. Ele possui uma arquitetura de 8 bits com CPU core baseado no núcleo STM8 de 24 MHz, e oferece uma ampla variedade de recursos, incluindo interface de comunicação serial, temporizadores, ADC, PWM e outros. Com 8 KB de memória flash programável, 1 KB de RAM e 640 bytes de memória EEPROM, o STM8S003F3P é uma opção econômica, pois atende muito bem este projeto que não exige processamento de alta complexidade, além do fabricante disponibilizar compilador (*Cosmic*) e IDE (*ST visual developer* - STVD) de forma gratuita no qual ele foi programado em linguagem C. (ST, 2018)

Através da comunicação serial o microcontrolador comunica ao *smartphone* o estado em que se encontra e os valores de medição obtidos. Utilizando os pinos de saída do circuito analógico, mede a tensão utilizando o AD de 10 bits e quantifica os valores em índices percentuais de glicose. Também controla o LED, para indicar ao usuário o *status* do dispositivo.

Na figura 11 é possível visualizar o circuito do microcontrolador. O capacitor C7 está disposto na configuração de desacoplamento e será posicionado na PCB o mais próximo possível do pino de alimentação.

Figura 11 – Fonte de alimentação com reguladores lineares.

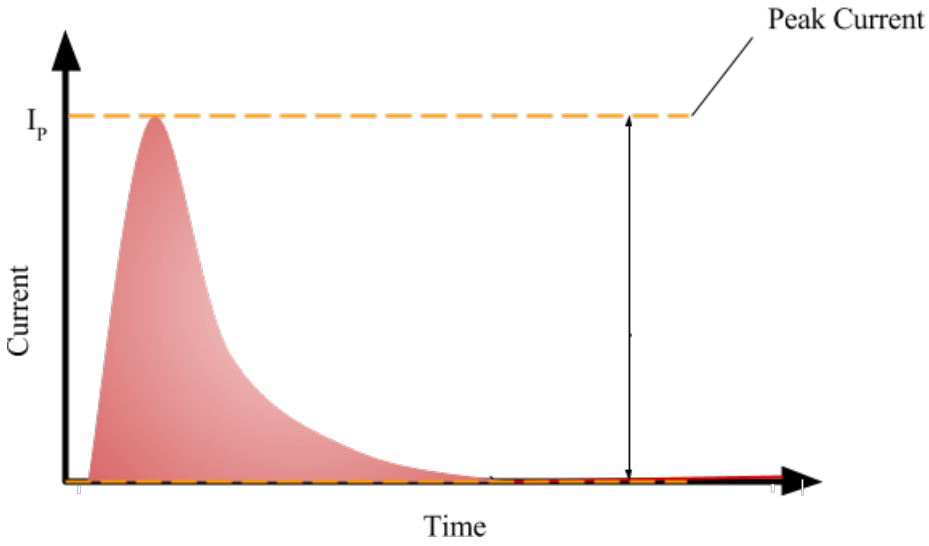


Fonte: Elaborado pelo autor (2023)

3.4 Condicionamento do sinal

O sinal esperado é não periódico e com um pico de corrente no início da captura. Após os instantes iniciais do pico, o sinal vai perdendo intensidade até se aproximar de zero em cerca de 15 segundos, em uma assíntota horizontal, podemos ver a ilustração do sinal característico na figura 12.

Figura 12 – Sinal característico da corrente no transdutor.

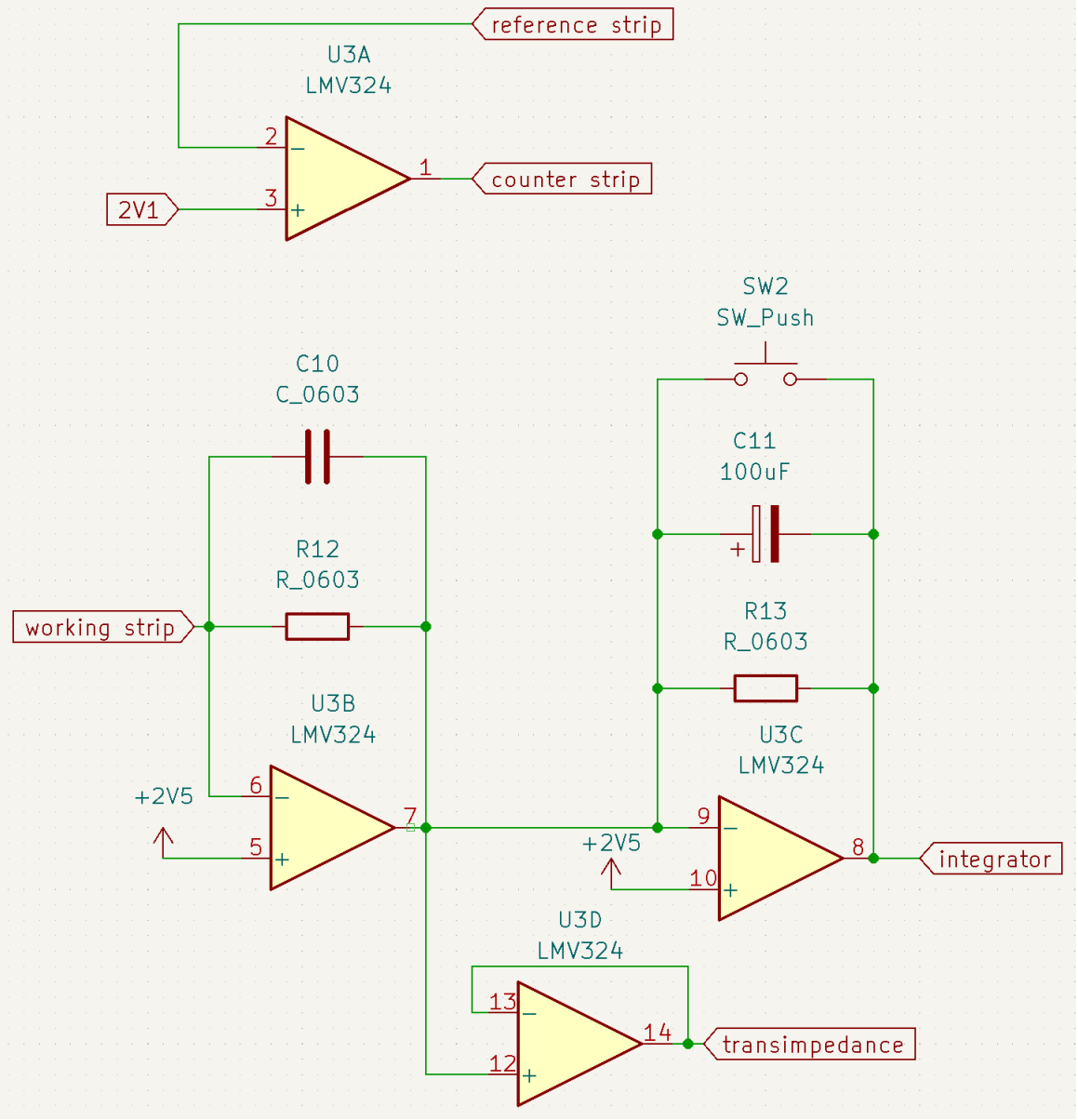


Fonte: (HEATH, 2017) - editado pelo autor (2023)

A hipótese inicial é de que é possível determinar a glicose com uma medida única de um ponto no tempo (YANEZ, 2013). Sendo assim, é necessário validar se a hipótese é verdadeira e, caso positivo, qual é o melhor tempo após o pico de corrente para realizar a captura da medição de corrente naquele ponto.

Na figura 13 é possível visualizar o circuito da interface de condicionamento do sinal das tiras de medição, onde é realizada a conversão do sinal de corrente para tensão, para ser medido no conversor AD do microcontrolador.

Figura 13 – Circuito de condicionamento do sinal das tiras de medição.



Fonte: Elaborado pelo autor (2023)

Os AOPs escolhidos foram do CI LMV324, que possui quatro amplificadores operacionais *rail to rail* independentes em um único invólucro, o que o torna conveniente para o projeto que necessita de ao menos 3 AOPs. A faixa de tensão de alimentação é de 2,7 a 5,5 volts, o que o torna adequado para a aplicação, tendo em vista que a alimentação do circuito é de 5V. Além disso, em caso de migração do circuito de alimentação principal para a tensão de 3.3V, o arranjo do circuito poderia ser mantido com apenas pequenos ajustes nos valores dos componentes discretos.

O LMV324 possui GBW de 1,2 MHz e uma margem de ganho de 10 dB, o que o torna adequado para aplicações que exigem boa precisão e resposta rápida — que não é o caso deste projeto, mas que dá margem a futuras modificações. Também possui uma corrente de polarização de entrada muito baixa, ajudando a reduzir o consumo de energia em aplicações de baixa potência. (TI, 1999)

3.4.1 Amplificador operacional de trans-impedância

O TIA é basicamente um amplificador operacional com um resistor de *feedback* conectado entre a saída e a entrada inversora. A entrada não inversora é geralmente mantida em um potencial de referência. O resistor de *feedback* é escolhido para definir a ganho do TIA e como a tensão de saída é diretamente proporcional à corrente de entrada, o ganho é dado pela razão entre a resistência de *feedback* e a resistência do sensor de corrente. Em suma, quando a corrente flui por meio de uma carga, é produzida uma queda de tensão. O TIA usa essa queda de tensão para produzir uma tensão de saída proporcional à corrente de entrada.

O AOP U3B da figura 13 está polarizado na configuração TIA e sua saída está ligada no AD do microcontrolador, sendo assim possível realizar a análise da corrente e estimar o percentual de glicose.

O procedimento de ajuste do TIA (U3B) para o resistor de *feedback* (R12) foi feito de maneira empírica, pois o valor máximo do pico de corrente ainda não era conhecido. Inicialmente foi disposto um resistor de 500KΩ e, ao realizar a medição houve saturação no canal AD, chegando a 5V na saída do TIA.

Ao colocar um resistor de 200KΩ, o pico de tensão na saída do TIA foi de 4,8V sendo assim, o resistor escolhido. Conhecendo a tensão e o valor do resistor de *feedback*, foi possível calcular o pico de corrente do transdutor, que é de 24μA, em uma DDP de 400mV.

3.4.2 Amplificador operacional integrador

O amplificador operacional integrador é um circuito eletrônico que usa um amplificador operacional para integrar um sinal de entrada ao longo do tempo. É constituído por

um amplificador operacional com um capacitor conectado entre sua entrada inversora e saída. A entrada não inversora é geralmente conectada no sinal de referência.

Quando um sinal de entrada é aplicado à entrada inversora, o capacitor começa a se carregar ou descarregar. A taxa de variação da tensão no capacitor é determinada pelo valor da resistência e do capacitor. Quanto maior o valor do capacitor, mais lenta será a taxa de variação da tensão na saída.

A saída do amplificador operacional integrador é proporcional à integral do sinal de entrada ao longo do tempo. Isso significa que, se o sinal de entrada for uma onda senoidal, a saída será uma onda cossenoidal, com um deslocamento de fase de 90 graus em relação ao sinal de entrada.

O AOP U3C da figura 13 está polarizado na configuração de integrador e sua saída está ligada no AD do microcontrolador, sendo assim possível realizar a análise da tensão ao final da medição e estimar a quantidade de glicose no sangue. Este AOP foi adicionado de maneira preventiva, caso a medida com o TIA não fosse satisfatória e não fosse possível indicar uma curva de tendência, falhando a hipótese inicial. Também poderia ser desprezado, tendo em vista que é possível realizar a integração do sinal de saída do amplificador de transimpedância no próprio microcontrolador caso necessário.

Em virtude dos resultados satisfatórios com o TIA, não foi necessário continuar o trabalho de dimensionamento dos componentes do integrador, a fim de achar sua constante de tempo.

3.4.3 Amplificador operacional de ganho unitário

O AOP (amplificador operacional) de ganho unitário é um tipo de amplificador operacional com um ganho de tensão próximo a 1, ou seja, não amplifica o sinal de entrada. Em outras palavras, a saída do AOP de ganho unitário é igual à entrada.

O AOP de ganho unitário é frequentemente usado como um *buffer* de sinal, sendo um circuito que isola o circuito do circuito de saída, evitando que uma impedância de carga afete o sinal de entrada. Ele é usado para aumentar a capacidade de carga de um sinal sem afetar o sinal original. O amplificador U3A irá atuar como *buffer* de sinal quando o sangue formar uma ponte eletrolítica entre o pino da entrada inversora e a saída.(WENDLING, 2010)

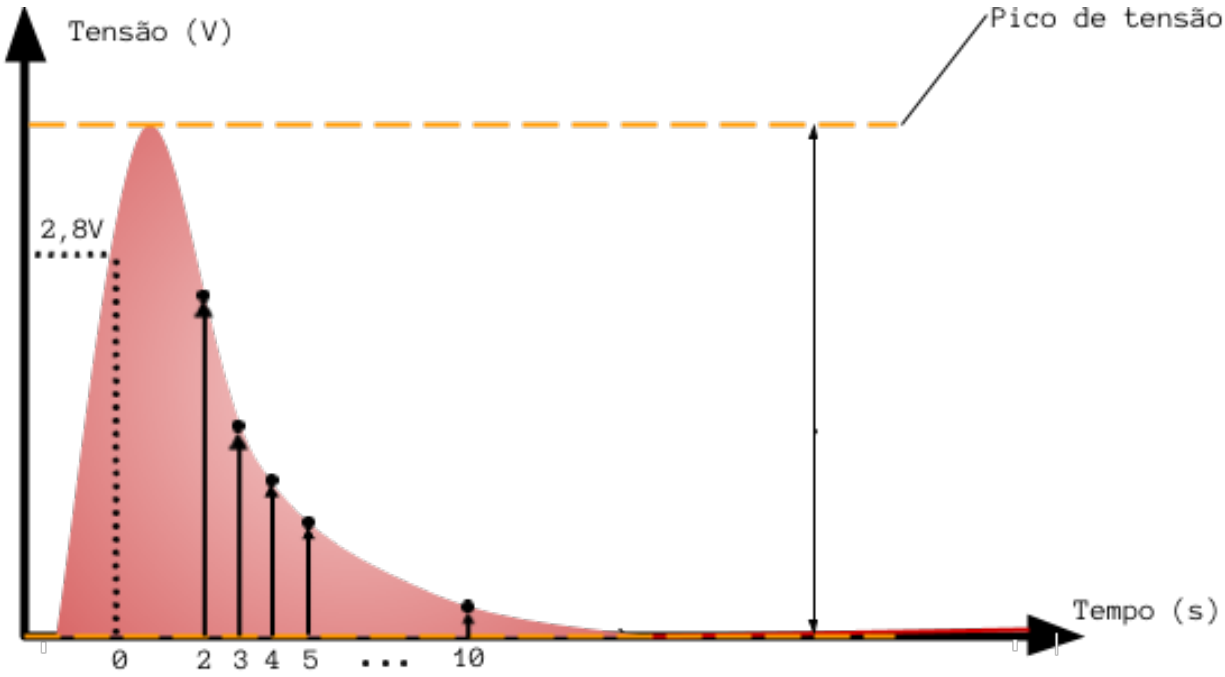
3.5 Regressão linear e modelo proposto

O sinal característico da curva de tensão da medida é um pico que chega próximo da saturação da saída do amplificador de transimpedância no momento inicial e, posteriormente, vai diminuindo a tensão gradativamente tendendo a zero. Para realizar a conversão da tensão medida pelo microcontrolador no transdutor, é necessário um modelo matemático onde seja possível entrar com o parâmetro de tensão e receber como saída o nível de glicose no sangue e, para isto, foi utilizada a técnica de regressão linear.

A coleta de dados para o cálculo do modelo matemático foi feita com dois voluntários diabéticos, pois o nível de glicose no sangue pode variar muito mais em relação aos não diabéticos. Foi realizado 19 capturas ao longo de 24 horas e em diversas situações, alcançando a expectativa de preencher a tabela de pares ordenados (glicose, tensão) com valores diversos, a fim de trazer um modelo que consiga representar todo o espectro de concentração de glicose no sangue.

Para determinar qual o melhor instante para realizar a captura do valor de após o a tensão ultrapassar o *thresold* de 2,8V, cada amostra foi medida 9 vezes de 2 a 10 segundos, com intervalos de 1 segundo. A figura 14 exemplifica a amostragem feita no sinal e a tabela 1 contém os dados.

Figura 14 – Amostragem do sinal para determinar o melhor tempo de captura.



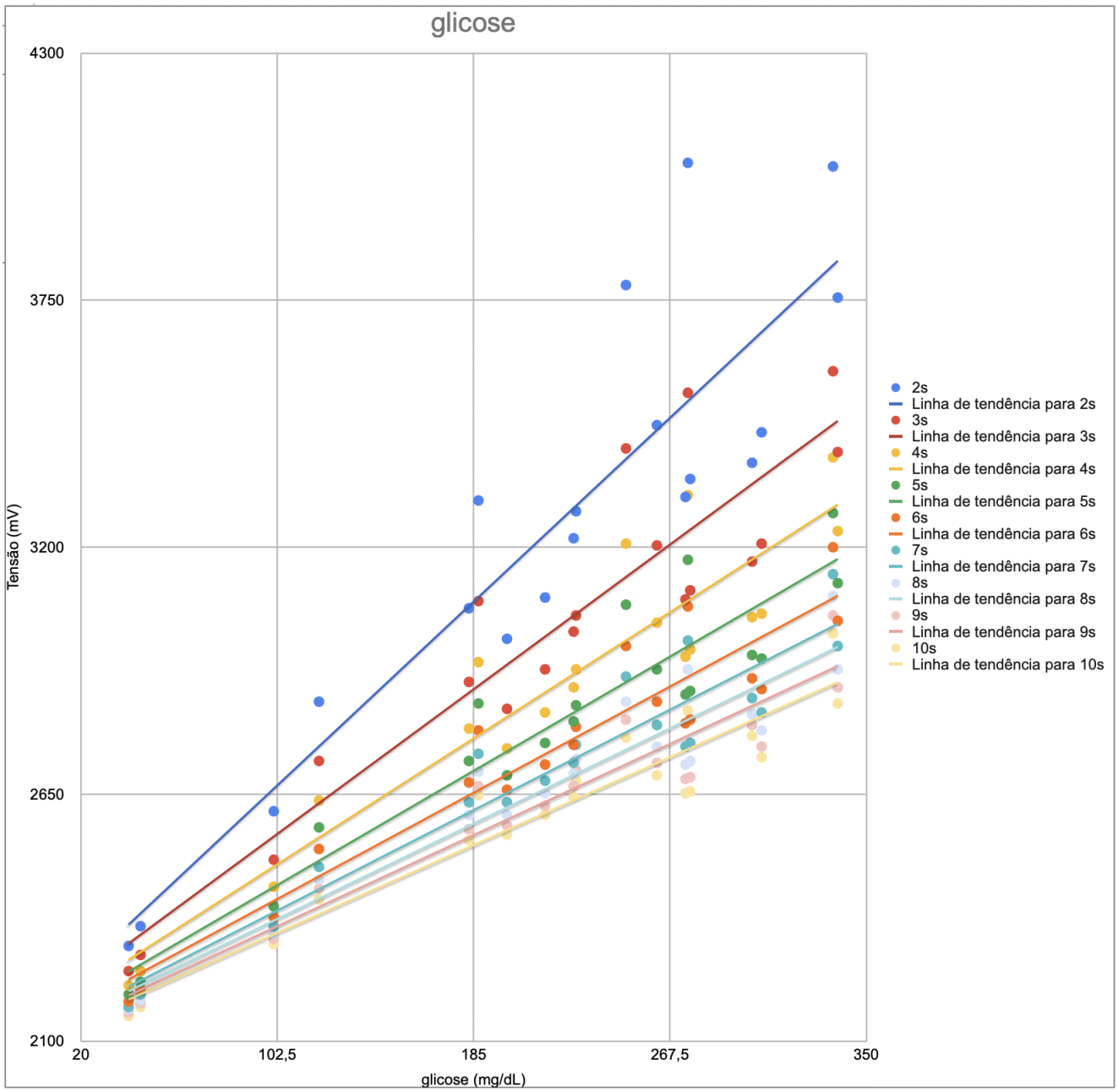
Fonte: (HEATH, 2017) - editado pelo autor (2023)

Tabela 1 – Tensão da medida em relação à glicose no aparelho de referência.

glicose (mg/dL)	**2s (mV)**	**3s (mV)**	**4s (mV)**	**5s (mV)**	**6s (mV)**	**7s (mV)**	**8s (mV)**	**9s (mV)**	**10s (mV)**
187	3304	3080	2944	2852	2792	2740	2700	2668	2648
276	3352	3104	2972	2880	2816	2764	2724	2688	2656
227	3220	3012	2888	2812	2760	2720	2696	2668	2644
215	3088	2928	2832	2764	2716	2680	2652	2624	2604
40	2312	2256	2224	2204	2188	2176	2172	2164	2156
45	2356	2292	2256	2232	2216	2204	2192	2184	2176
101	2612	2504	2444	2400	2376	2356	2340	2328	2316
274	3312	3084	2956	2872	2808	2756	2716	2684	2652
199	2996	2840	2752	2692	2660	2632	2604	2580	2560
228	3280	3048	2928	2848	2800	2760	2728	2704	2680
275	4056	3544	3316	3172	3068	2992	2928	2876	2836
302	3388	3168	3044	2960	2908	2864	2828	2804	2780
338	3756	3412	3236	3120	3036	2980	2928	2888	2852
262	3472	3204	3032	2928	2856	2804	2756	2720	2692
306	3456	3208	3052	2952	2884	2832	2792	2756	2732
249	3784	3420	3208	3072	2980	2912	2856	2816	2776
336	4048	3592	3400	3276	3200	3140	3092	3048	3008
183	3064	2900	2796	2724	2676	2632	2604	2572	2548
120	2856	2724	2636	2576	2528	2488	2460	2440	2416

A técnica de regressão linear foi aplicada aos dados coletados para cada medida após intervalo de tempo, a fim de encontrar a equação que melhor descreve a relação entre as variáveis e o melhor tempo para a captura da medida. A equação de regressão linear é representada por y = mx + b, onde "y"é a concentração de glicose, "x"é a tensão, "m"é o coeficiente angular (inclinação da reta) e "b"é o coeficiente linear (intercepto). O resultado de cada linha de tendência é evidenciado na figura 15.

Figura 15 – gráfico com linhas de tendência para cada série de tempo.

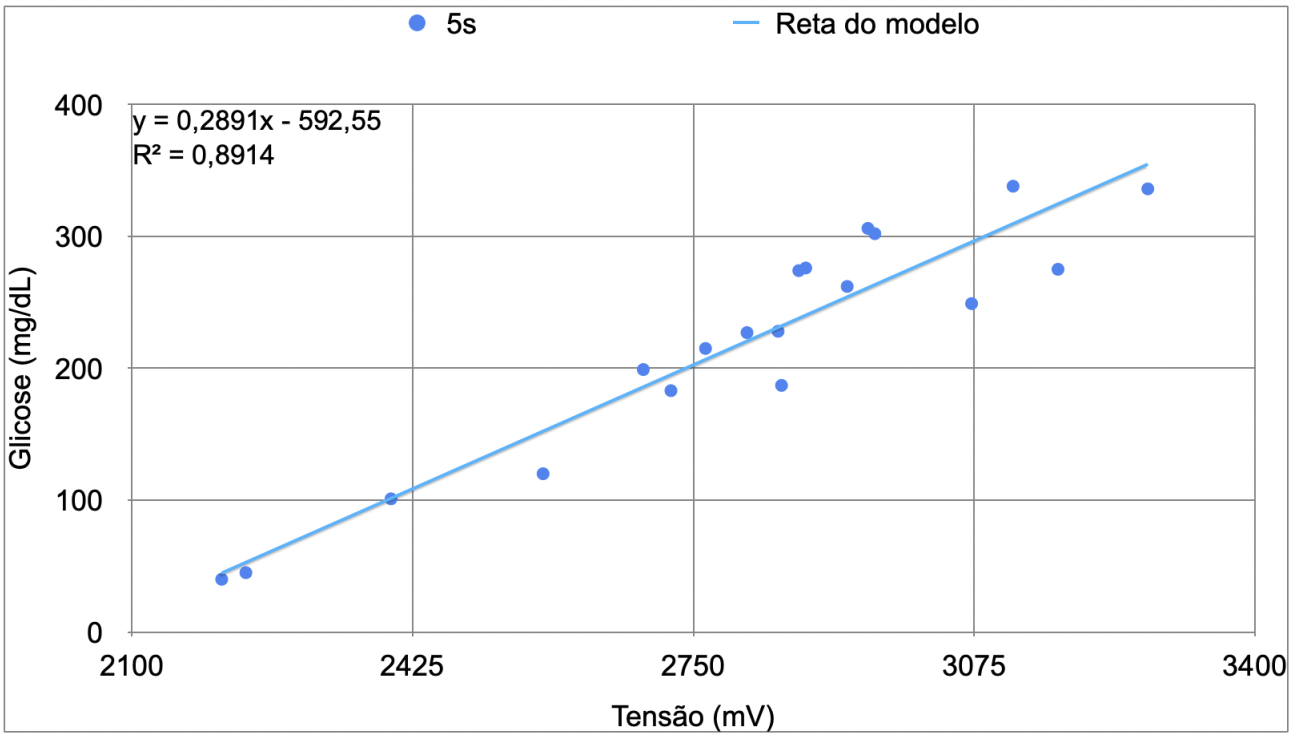


Fonte: Elaborado pelo autor (2023)

Após a análise, o modelo que se apresentou mais robusto foi o de 5 segundos, pois apresenta um R^2 de 0,891 e, a curva de tendência em sua saturação máxima, permitirá

medir taxas de glicose no sangue de até 853 mg/dL. Na figura 16 é possível verificar o modelo e o R^2.

Figura 16 – gráfico com linhas de tendência para a série de 5 segundos.

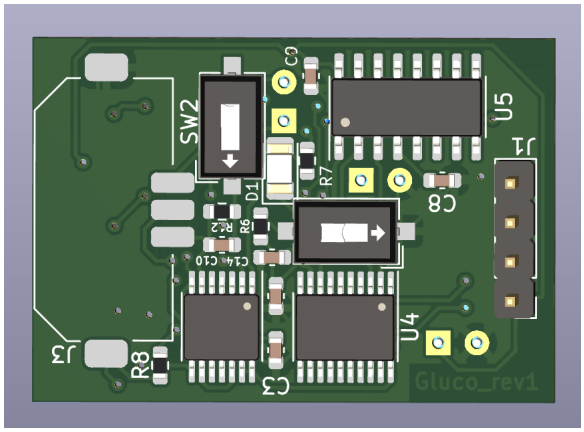


Fonte: Elaborado pelo autor (2023)

3.6 Placa de circuito impresso

O software utilizado para o desenvolvimento do esquemático e do layout da PCB é o *Kicad*, um software livre e de código aberto para design de circuitos eletrônicos e criação de placas de circuito impresso. Na figura 17 é possível visualizar a placa de circuito impresso em renderização 3d.

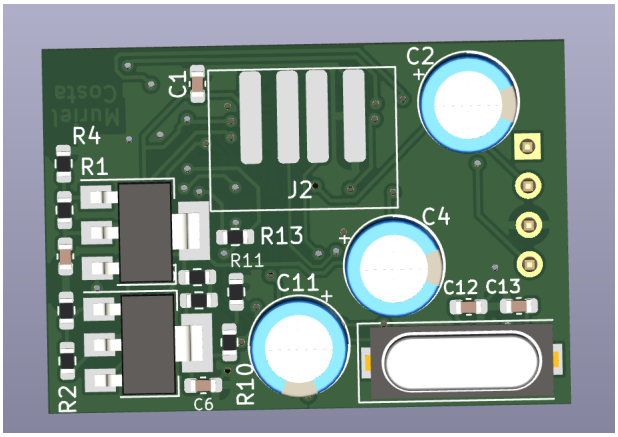
Figura 17 – Frente da placa de circuito impresso em renderização 3d.



Fonte: Elaborado pelo autor (2023)

A placa possui 2 camadas de cobre separadas por uma camada dielétrica de 1,6 milímetros de espessura. A dimensão final é de 33 milímetros de largura e 23 milímetros de altura. Na figura 18 é possível visualizar que ficou posicionado a maioria do circuito da fonte de alimentação, enquanto na figura 17 estão a maioria dos componentes lógicos do circuito. Isto se dá pela necessidade em minimizar possíveis problemas de ruído no circuito de aquisição do sinal.

Figura 18 – Verso da placa de circuito impresso em renderização 3d.



Fonte: Elaborado pelo autor (2023)

3.7 Aplicativo de interface

O Aplicativo de *smartphone* deste projeto foi desenvolvido utilizando tecnologia *React Native*. O *React Native* utiliza uma abordagem inovadora que permite aos desenvolvedores escrever código em *JavaScript*, aproveitando os princípios da biblioteca *React*, e produzir aplicativos nativos que podem ser executados tanto em dispositivos *iOS* quanto *Android*, evitando que se mantenha duas bases de código, uma para cada sistema operacional.

A estilização construída utilizando elementos de CSS e HTML, que permite uma maior produtividade para desenvolvedores já familiarizados com desenvolvimento de páginas WEB. No roteamento das páginas foi utilizado a biblioteca *React Navigation* e a navegação entre as telas usando *stack navigators* como estrutura de rota. Para comunicação serial com o hardware foi utilizado a biblioteca de terceiro *react-native-usb-serialport*.

A lógica de comunicação serial foi implementada no aplicativo, estabelecendo conexões, enviando e recebendo dados e processando as respostas, conforme as especificações do dispositivo e do protocolo de comunicação acordado entre a interface e o periférico.

Além dessas etapas, também foi necessário lidar com outras tarefas típicas de desenvolvimento de aplicativos, como gerenciamento de estado, onde foi utilizado o recurso nativo do *hook useState*.

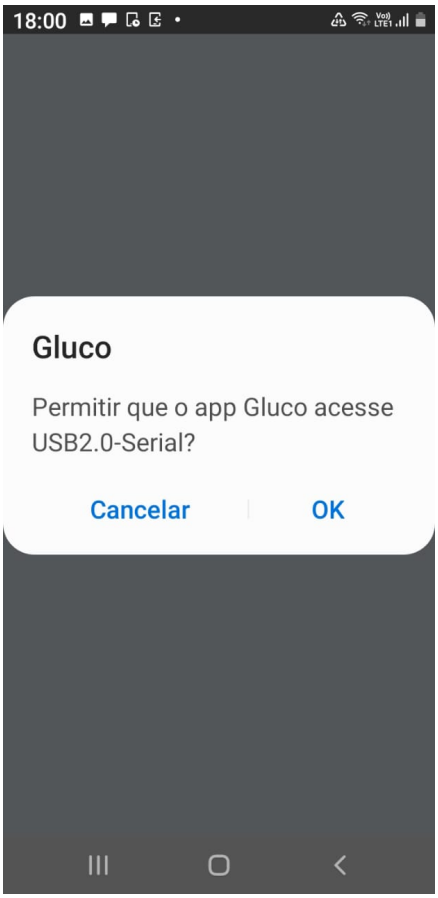
Na figura 19 é possível visualizar a tela inicial do aplicativo, onde pede para o usuário inserir o dispositivo medidor de glicose.

Figura 19 – Tela inicial do aplicativo.

Fonte: Elaborado pelo autor (2023)

No momento em que o dispositivo é inserido, é solicitada a autorização de uso da comunicação serial, conforme figura 20.

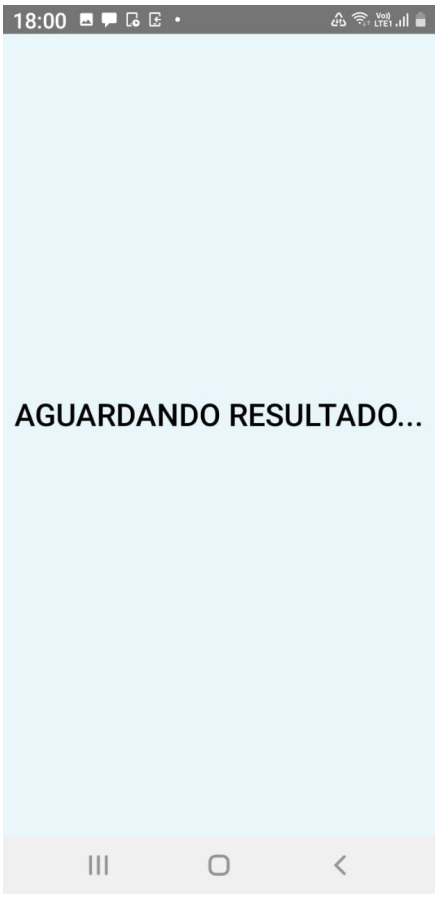
Figura 20 – Solicitação de autorização do uso da comunicação serial do celular.



Fonte: Elaborado pelo autor (2023)

Após autorizar o uso da porta serial, é mostrada a mensagem de "aguardando resultado", onde o sistema fica à espera da inserção da fita de medição com sangue, conforme figura 21. Uma vez inserida a amostra de sangue, o sistema pede para o usuário aguardar e em seguida mostra o valor do resultado da medição.

Figura 21 – Aplicativo aguardando o resultado da medição de glicose.



Fonte: Elaborado pelo autor (2023)

4 Conclusão

O presente estudo representa um passo fundamental na pesquisa e desenvolvimento de um medidor de glicose moderno e de menor custo. O objetivo principal deste projeto foi projetar, calibrar e validar um medidor de glicose que possa fornecer medições dos níveis de glicose no sangue, tornando-o uma ferramenta valiosa no monitoramento e tratamento de pacientes com diabetes mellitus e outras condições relacionadas à glicose.

Durante a execução deste projeto, foram realizadas várias etapas críticas, incluindo a coleta de dados de referência através de medidor de glicose comercial, a aquisição de medidas de tensão pelo medidor de glicose em conjunto com as medidas de glicose de referência e a aplicação da análise de regressão linear para estabelecer uma relação matemática entre as medidas de tensão e os níveis reais de glicose no sangue. Os resultados obtidos indicam que o medidor de glicose foi calibrado com sucesso, proporcionando a conversão das medidas de tensão em leituras de glicose.

A calibração bem-sucedida deste medidor de glicose tem implicações significativas para a área da saúde, uma vez que a monitorização eficaz dos níveis de glicose é essencial para o tratamento e gestão adequados de pacientes com diabetes e outras condições metabólicas. A capacidade de obter medições de glicose precisas em tempo real pode melhorar a qualidade de vida dos pacientes, prevenir complicações associadas à hiperglicemia ou hipoglicemia e proporcionar informações cruciais para profissionais de saúde.

É importante destacar que, embora tenhamos alcançado resultados promissores neste projeto, a pesquisa e o desenvolvimento contínuos são necessários para aprimorar ainda mais a precisão e a confiabilidade do medidor de glicose. Além disso, é essencial conduzir estudos clínicos adicionais para avaliar a eficácia e a segurança do dispositivo em uma ampla variedade de cenários clínicos. Porém o sucesso da calibração e a validação preliminar do dispositivo são motivos de otimismo para o desenvolvimento futuro e a eventual disponibilidade de um medidor de glicose altamente confiável, acessível e moderno que pode melhorar substancialmente a qualidade de vida dos pacientes com diabetes e contribuir para o avanço da medicina e da saúde pública.

Referências

ACCUMED. *Fabricante*. 2023. Disponível em: <www.accumed.com.br>. Acesso em: 27 de setembro 2023. Citado na página 3.

ANALOG DEVICES. *Decoupling Techniques*. [S.l.], 2009. Disponível em: <https://www.analog.com/media/en/training-seminars/tutorials/MT-101.pdf>. Citado na página 14.

BANKAR MAHESH V. BULE, R. S. S. S. B.; ANANTHANARAYAN, L. Glucose oxidase — an overview. v. 27, p. 489–501, 2009. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0734975009000536>. Acesso em: 17 outubro 2023. Citado na página 7.

BRASIL, M. da S. *Manual de Hipertensão Arterial e Diabetes Mellitus*. Brasília: Ministério da Saúde: [s.n.], 2002. Citado 2 vezes nas páginas 1 e 2.

DREAMCITY INNOVATIONS. *CH340G USB to UART Interface Datasheet*. [S.l.], 2014. Disponível em: <https://static.efetividade.net/img/ch340g-datasheet-34852.pdf>. Citado 2 vezes nas páginas 11 e 12.

HEATH, J. *What is inrush current?* [S.l.], 2017. Disponível em: <https://www.powerelectronictips.com/what-is-inrush-current/>. Citado 2 vezes nas páginas 17 e 21.

INTERNATIONAL DIABETES FEDERATION. *IDF Diabetes Atlas*. Brussels, Belgium, 2021. Disponível em: <https://www.diabetesatlas.org>. Citado na página 1.

MICROCHIP TECHNOLOGY INC. *Glucose Meter Reference Design*. [S.l.], 2013. Disponível em: <https://ww1.microchip.com/downloads/cn/AppNotes/cn566535.pdf>. Citado 2 vezes nas páginas 5 e 7.

ONSEMI. *LM317 - Voltage Regulator – Adjustable Output, Positive 1.5 A*. [S.l.], 2021. Disponível em: <https://www.onsemi.com/download/data-sheet/pdf/lm317-d.pdf>. Citado na página 14.

STMICROELECTRONICS. *STM8S003F3 STM8S003K3*. [S.l.], 2018. Disponível em: <https://www.st.com/resource/en/datasheet/stm8s003f3.pdf>. Citado na página 15.

SUNROM.COM. *MicroUSB pinout*. [S.l.], 2022. Disponível em: <hwww.sunrom.com>. Citado na página 13.

TEXAS INSTRUMENTS. *LMV3xx Low-Voltage Rail-to-Rail Output Operational Amplifier*. [S.l.], 1999. Disponível em: <https://www.ti.com/lit/ds/symlink/lmv324.pdf>. Citado na página 19.

TEXAS INSTRUMENTS. *Battery chargers in USB OTG devices*. [S.l.], 2010. Disponível em: <https://www.ti.com/lit/wp/sszy001/sszy001.pdf>. Citado na página 13.

WENDLING, M. *Amplificadores operacionais*. [S.l.], 2010. Disponível em: <https://www.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/3---amplificadores-operacionais-v2.0.pdf>. Citado na página 20.

WILSON, R.; TURNER, A. Glucose oxidase: an ideal enzyme. v. 7, p. 165–185, 1992. Disponível em: <https://www.sciencedirect.com/science/article/pii/095656639287013F>. Acesso em: 16 outubro 2023. Citado na página 7.

YANEZ, M. G. *Glucose Meter Fundamentals and Design*. [S.l.], 2013. Disponível em: <https://www.nxp.com/docs/en/application-note/AN4364.pdf>. Citado na página 17.

Apêndices

APÊNDICE A – Código fonte do firmware

```
/* MAIN.C file
 *
 *  Medidor de glicose
 *
 *  Projeto de conclusão de curso de
 *  engenharia de informação.
 *
 *  Universidade federal do ABC.
 *
 * Com muito carinho
 * Muriel Costa
 * 2023
 */

#include "stm8s.h"
#include "stm8s_adc1.h"
#include "stm8s_uart1.h"
#include "main.h"
#include <stdio.h>
#include <string.h>

/*******************************************************************************
* Declaração de variáveis globais:                                             *
*******************************************************************************/
static union{
uint8_t c[2];
uint16_t a;
}adc_value;

uint16_t  adc_counter = 0;
uint16_t  bt_debounce_counter = 0;
uint16_t max_adc = 0;
uint16_t adc_timer_counter = 0;
uint16_t  adc_values_array[] = {0, 0, 0, 0, 0};
uint16_t  sum[] = {0, 0, 0, 0, 0};
```

```
uint8_t   adc_channel = 0;
uint8_t   adc_sample = 0;
uint8_t   tamanho_buffer = 0;
uint8_t   button = 0;
uint8_t  *tx_buffer = (uint8_t *) 0;
uint8_t  *aux = (uint8_t *) 0;
uint8_t   adc_uint8_array[] = {0, 0};
uint8_t   meassuring = FALSE;
uint8_t   measurement_state = 0;

char  str[20];
char lower_string_value[2];
char higher_string_value[2];
char measurement_string_value_array[5];

void timerInit(void);
void ioInit(void);
void adcInit(void);
void adc_update(void);
void uartInit(void);
void input_pins_scan (void);
void tx_uart (uint8_t *buffer, uint8_t quantidade_bytes);
void send_message(void);
void get_max_value_adc(void);
void start_conversion(void);
void led_turn_off(void);
void led_turn_on(void);
void clear_str_string(void);

main(){
CLK->CKDIVR = 0;
timerInit();
ioInit();
adcInit();
uartInit();
enableInterrupts();
while (1){
adc_update();
```

```
input_pins_scan();
//send_message();
get_max_value_adc();
start_conversion();
}
}
void get_max_value_adc(){
if(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] > max_adc){
max_adc = adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL];
}

}

void clear_str_string(){
int counter;
for(counter=0; counter < 15; counter++){
str[counter]='\0';
}
}

void start_conversion(){
int i=0;
int final = 3;
char temp;
if(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] > _2600mV || meassuring ){
meassuring = TRUE;
/*
if (
(adc_timer_counter > 2000 && adc_timer_counter < 3000)
&& measurement_state == 0){

led_turn_off();

i=0;
final = 3;
temp;
while(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] > 0){
measurement_string_value_array[i] =
(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] % 10) + '0';
```

```
adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] /= 10;
i++;
}
i=0;
while(i<final){
temp = measurement_string_value_array[i];
measurement_string_value_array[i] = measurement_string_value_array[final];
measurement_string_value_array[final] = temp;
i++;
final--;
}
str[0]='\0';
strcat(str, "|2s: " );
strcat(str, &measurement_string_value_array[0]);
strcat(str, "mV" );
tx_uart(&str[0], 11);
measurement_state++;



}
if (
(adc_timer_counter > 3000 && adc_timer_counter < 4000)
&& measurement_state == 1){

led_turn_on();

i=0;
final = 3;
temp;

while(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] > 0){
measurement_string_value_array[i] =
(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] % 10) + '0';
adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] /= 10;
i++;
}
i=0;
while(i<final){
temp = measurement_string_value_array[i];
```

```
measurement_string_value_array[i] = measurement_string_value_array[final];
measurement_string_value_array[final] = temp;
i++;
final--;
}
str[0]='\0';
strcat(str, "|3s: " );
strcat(str, &measurement_string_value_array[0]);
strcat(str, "mV" );
tx_uart(&str[0], 11);

measurement_state++;
}

if (
(adc_timer_counter > 4000 && adc_timer_counter < 5000)
&& measurement_state == 2){

led_turn_off();

i=0;
final = 3;
temp;

while(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] > 0){
measurement_string_value_array[i] =
(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] % 10) + '0';
adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] /= 10;
i++;
}
i=0;
while(i<final){
temp = measurement_string_value_array[i];
measurement_string_value_array[i] = measurement_string_value_array[final];
measurement_string_value_array[final] = temp;
i++;
final--;
}
str[0]='\0';
```

```
strcat(str, "|4s: " );
strcat(str, &measurement_string_value_array[0]);
strcat(str, "mV" );
tx_uart(&str[0], 11);
measurement_state++;



}
*/
if (
(adc_timer_counter > 5000 && adc_timer_counter < 6000)
){
led_turn_on();

i=0;
final = 3;
temp;
adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] =
0.2891*adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL]-592.55;

while(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] > 0){
measurement_string_value_array[i] =
(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] % 10) + '0';
adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] /= 10;
i++;
}
final = i+1;
i=0;
while(i<final){
temp = measurement_string_value_array[i];
measurement_string_value_array[i] = measurement_string_value_array[final];
measurement_string_value_array[final] = temp;
i++;
final--;
}
//measurement_string_value_array[i] = '.'
str[0]='\0';
strcat(str, &measurement_string_value_array[i]);
```

```
strcat(str, " mg/dL" );
tx_uart(&str[0], 10);
meassuring = FALSE;

}

/*
if (
(adc_timer_counter > 6000 && adc_timer_counter < 7000)
&& measurement_state == 4){
led_turn_on();

i=0;
final = 3;
temp;
while(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] > 0){
measurement_string_value_array[i] =
(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] % 10) + '0';
adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] /= 10;
i++;
}
i=0;
while(i<final){
temp = measurement_string_value_array[i];
measurement_string_value_array[i] = measurement_string_value_array[final];
measurement_string_value_array[final] = temp;
i++;
final--;
}
str[0]='\0';
strcat(str, "|6s: " );
strcat(str, &measurement_string_value_array[0]);
strcat(str, "mV" );
tx_uart(&str[0], 11);

measurement_state++;
}

if (
```

```
(adc_timer_counter > 7000 && adc_timer_counter < 8000)
&& measurement_state == 5){
led_turn_on();

i=0;
final = 3;
temp;
while(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] > 0){
measurement_string_value_array[i] =
(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] % 10) + '0';
adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] /= 10;
i++;
}
i=0;
while(i<final){
temp = measurement_string_value_array[i];
measurement_string_value_array[i] = measurement_string_value_array[final];
measurement_string_value_array[final] = temp;
i++;
final--;
}
str[0]='\0';
strcat(str, "|7s: " );
strcat(str, &measurement_string_value_array[0]);
strcat(str, "mV" );
tx_uart(&str[0], 11);

measurement_state++;
}

if (
(adc_timer_counter > 8000 && adc_timer_counter < 9000)
&& measurement_state == 6){
led_turn_on();

i=0;
final = 3;
temp;
while(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] > 0){
```

```
measurement_string_value_array[i] =
(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] % 10) + '0';
adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] /= 10;
i++;
}
i=0;
while(i<final){
temp = measurement_string_value_array[i];
measurement_string_value_array[i] = measurement_string_value_array[final];
measurement_string_value_array[final] = temp;
i++;
final--;
}
str[0]='\0';
strcat(str, "|8s: " );
strcat(str, &measurement_string_value_array[0]);
strcat(str, "mV" );
tx_uart(&str[0], 11);

measurement_state++;
}

if (
(adc_timer_counter > 9000 && adc_timer_counter < 10000)
&& measurement_state == 7){
led_turn_on();

i=0;
final = 3;
temp;
while(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] > 0){
measurement_string_value_array[i] =
(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] % 10) + '0';
adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] /= 10;
i++;
}
i=0;
while(i<final){
temp = measurement_string_value_array[i];
```

```
measurement_string_value_array[i] = measurement_string_value_array[final];
measurement_string_value_array[final] = temp;
i++;
final--;
}
str[0]='\0';
strcat(str, "|9s: " );
strcat(str, &measurement_string_value_array[0]);
strcat(str, "mV" );
tx_uart(&str[0], 11);

measurement_state++;
}

if (
(adc_timer_counter > 10000)
&& measurement_state == 8){

led_turn_on();

i=0;
final = 3;
temp;
while(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] > 0){
measurement_string_value_array[i] =
(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] % 10) + '0';
adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] /= 10;
i++;
}
i=0;
while(i<final){
temp = measurement_string_value_array[i];
measurement_string_value_array[i] = measurement_string_value_array[final];
measurement_string_value_array[final] = temp;
i++;
final--;
}
str[0]='\0';
strcat(str, "|10s: " );
```

```
strcat(str, &measurement_string_value_array[0]);
strcat(str, "mV" );
tx_uart(&str[0], 12);

measurement_state=0;


meassuring = FALSE;
}
*/
}else{
adc_timer_counter = 0;
led_turn_on();
}
}

void send_message(void){

int i=0;
int final = 3;
char temp;
if (button){
while(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] > 0){
measurement_string_value_array[i] =
(adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] % 10) + '0';
adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] /= 10;
i++;
}
i=0;

while(i<final){
temp = measurement_string_value_array[i];
measurement_string_value_array[i] = measurement_string_value_array[final];
measurement_string_value_array[final] = temp;
i++;
final--;
}

strcat(str, "|1s: " );
```

```
strcat(str, &measurement_string_value_array[0]);
strcat(str, "mV" );

tx_uart(&str[0], 11);
button = 0;
}
}


/*****************************************************************************
 * input_pins_scan();                                                        *
 *                                                                           *
 * Procedimento de checagem e debounce do estado dos  botões;                *
 *                                                                           *
 * void -> não recebe parâmetros;                                            *
 *                                                                           *
 * void -> não retorna valor;                                                *
 *****************************************************************************/
void input_pins_scan (void){

if(!GPIO_ReadInputPin(BUTTON_PIN)){

if(bt_debounce_counter >= TIME_DEBOUNCE_BT ){

button = TRUE;
meassuring = TRUE;
bt_debounce_counter = 0;
}

}else{

if(bt_debounce_counter >= TIME_DEBOUNCE_BT){

button = FALSE;
bt_debounce_counter = 0;
}
}
}
```

```
void led_turn_on(){
GPIO_WriteLow(LED_PIN);
}

void led_turn_off(){
GPIO_WriteHigh(LED_PIN);
}

/*******************************************************************************
 * inicializa_uart();                                                          *
 *                                                                             *
 * Procedimento de inicialização da comunicação uart;                          *
 *                                                                             *
 * void -> não recebe parâmetros;                                              *
 *                                                                             *
 * void -> não retorna valor                                                   *
 *******************************************************************************/
void uartInit(void){
UART1_DeInit();
UART1_Init(
BAUD_RATE,
DATA_BITS,
STOP_BITS,
PARITY,
UART1_SYNCMODE_CLOCK_DISABLE,
UART1_MODE_TX_ENABLE
); //UART1_MODE_TXRX_ENABLE
UART1_ITConfig(UART1_IT_RXNE, DISABLE);
UART1_ITConfig(UART1_IT_TXE, DISABLE);
UART1_Cmd(ENABLE);
}

/*******************************************************************************
 * tx_uart(&buffer, quantidade_bytes);                                         *
 *                                                                             *
 * Procedimento de envio de dados pela uart;                                   *
 *                                                                             *
 * void -> não recebe parâmetros;                                              *
 *                                                                             *
```

```
 * void -> não retorna valor                                                  *
 ******************************************************************************/
void tx_uart(uint8_t *buffer, uint8_t quantidade_bytes){
tx_buffer = buffer;
tamanho_buffer = quantidade_bytes;
UART1_SendData8(*tx_buffer++);
tamanho_buffer--;

if(tamanho_buffer > 0){
UART1_ITConfig(UART1_IT_TXE, ENABLE);
}

}

/*******************************************************************************
* timerInit();                                                                 *
*                                                                              *
* Procedimento de configuração de Timer                                        *
*                                                                              *
* void -> não recebe parâmetros;                                               *
*                                                                              *
* void -> não retorna valor;                                                   *
*                                                                              *
* Configurações de Timer:                                                      *
*                                                                              *
* Timer 4:                                                                     *
* Prescaler = 64                                                               *
* Overflow  = 250                                                *
*                                                                             *
* T = PRESCALER * OVERFLOW / FREQUENCIA DE CLOCK                               *
*                                                                              *
* T = 64 * 250 / 16.000.000 = 1ms                                              *
*                                                                              *
* Timer 2:                                                                     *
* Prescaler = 64                                                               *
* Overflow  = 210                                                              *
*                                                                              *
* T = PRESCALER * OVERFLOW / FREQUENCIA DE CLOCK                               *
*                                                                              *
```

```
* T = 16 * 210 / 16.000.000 = 0,21ms          *
*  *
*  *
* Timer 4:                                                                    *
* Prescaler = 8                                                               *
* Overflow  = 16                                          *
*                                                         *
* T = PRESCALER * OVERFLOW / FREQUENCIA DE CLOCK                              *
*                                                                             *
* T = 6 * 16 / 128.000 = 1ms                                                *
*      *
*******************************************************************************/
void timerInit(void){
TIM4_DeInit      ();
TIM4_TimeBaseInit(TIM4_PRESCALER_64, 249);
TIM4_ITConfig    (TIM4_IT_UPDATE, ENABLE);
TIM4_Cmd         (ENABLE);
}

/*******************************************************************************
 * void ioInit(void);                                                          *
 *                                                                             *
 * Inicialização dos IO's                                                      *
 *                                                                             *
 *******************************************************************************/
void ioInit(void){
GPIO_Init(BUTTON_PIN, GPIO_MODE_IN_PU_NO_IT);
GPIO_Init(LED_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);
GPIO_WriteLow(LED_PIN); //LED TURN ON
}

/*******************************************************************************
 * adcInit();                                                                  *
 *                                                                             *
 * Procedimento de inicialização do conversor analógico/digital;               *
 *                                                                             *
 * void -> não recebe parâmetros;                                              *
 *                                                                             *
 * void -> não retorna valor                                                   *
```

```
 ***************************************************************************/
void adcInit(void){
ADC1_ConversionConfig(ADC1_CONVERSIONMODE_SINGLE, ADC1_CHANNEL_4, ADC1_ALIGN_RIGHT);

ADC1_PrescalerConfig (ADC1_PRESSEL_FCPU_D2);
ADC1_DataBufferCmd  (ENABLE);
ADC1_ScanModeCmd     (ENABLE);
ADC1_Cmd             (ENABLE);
}

/***************************************************************************
 * atualiza_adc();                                                         *
 *                                                                         *
 * Procedimento de leitura das entradas analógicas;                        *
 *                                                                         *
 * void -> não recebe parâmetros;                                          *
 *                                                                         *
 * void -> não retorna valor                                               *
 ***************************************************************************/
void adc_update(void){

 if(adc_counter >= SAMPLING_TIME){

 adc_counter = 0;
 ADC1_StartConversion();
   while(!ADC1_GetITStatus(ADC1_IT_EOC)){}
 ADC1_ClearFlag(ADC1_IT_EOC);
 aux = (uint8_t *)&ADC1->DBORH;

 for(adc_channel = 0; adc_channel < CHANNEL_QTY; adc_channel++){

 adc_value.c[0] = *aux;
 adc_value.c[1] = *(aux + 1);
 sum[adc_channel] += adc_value.a;

       aux = aux + 2;

  }
```

```
 adc_sample++;

 if(adc_sample == SAMPLE_QTY){

adc_sample = 0;


adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] =
sum[TRANSIMPEDANCE_ADC_CHANNEL] / SAMPLE_QTY;
adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] =
adc_values_array[TRANSIMPEDANCE_ADC_CHANNEL] * 4.888;

sum[TRANSIMPEDANCE_ADC_CHANNEL] = 0;


  }
    }
  }
```

APÊNDICE B – Código fonte do aplicativo

```
import React, {Component} from 'react';

import {
  StyleSheet,
  Text,
  View,
  TextInput,
  TouchableOpacity,
  ScrollView,
  Alert,
  DeviceEventEmitter,
} from 'react-native';
import {RNSerialport, definitions, actions} from 'react-native-serialport';
import {Container, InstructionText, InsertDeviceWrapper, Input}
from './index.style';
import SmartphoneSVG from '../../assets/cellphone-svgrepo-com.svg';
import UsbDeviceSVG from '../../assets/usb-svgrepo-com.svg';
import ArrowUpSVG from '../../assets/arrow-up-mark-svgrepo-com.svg';

class ManualConnection extends Component {
  constructor(props) {
    super(props);

    this.state = {
      servisStarted: false,
      connected: false,
      usbAttached: false,
      output: '',
      outputArray: [],
      baudRate: '115200',
      interface: '-1',
      sendText: 'HELLO',
      returnedDataType: definitions.RETURNED_DATA_TYPES.HEXSTRING,
    };
```

```
    this.startUsbListener = this.startUsbListener.bind(this);
    this.stopUsbListener = this.stopUsbListener.bind(this);
  }

  componentDidMount() {
    this.startUsbListener();
  }

  componentWillUnmount() {
    this.stopUsbListener();
  }

  startUsbListener() {
    DeviceEventEmitter.addListener(
      actions.ON_SERVICE_STARTED,
      this.onServiceStarted,
      this,
    );
    DeviceEventEmitter.addListener(
      actions.ON_SERVICE_STOPPED,
      this.onServiceStopped,
      this,
    );
    DeviceEventEmitter.addListener(
      actions.ON_DEVICE_ATTACHED,
      this.onDeviceAttached,
      this,
    );
    DeviceEventEmitter.addListener(
      actions.ON_DEVICE_DETACHED,
      this.onDeviceDetached,
      this,
    );
    DeviceEventEmitter.addListener(actions.ON_ERROR, this.onError, this);
    DeviceEventEmitter.addListener(
      actions.ON_CONNECTED,
      this.onConnected,
      this,
    );
```

```
  DeviceEventEmitter.addListener(
    actions.ON_DISCONNECTED,
    this.onDisconnected,
    this,
  );
  DeviceEventEmitter.addListener(actions.ON_READ_DATA, this.onReadData, this);
  RNSerialport.setReturnedDataType(this.state.returnedDataType);
  RNSerialport.setAutoConnectBaudRate(parseInt(this.state.baudRate, 10));
  RNSerialport.setInterface(parseInt(this.state.interface, 10));
  RNSerialport.setAutoConnect(true);
  RNSerialport.startUsbService();
}

stopUsbListener = async () => {
  DeviceEventEmitter.removeAllListeners();
  const isOpen = await RNSerialport.isOpen();
  if (isOpen) {
    Alert.alert('isOpen', isOpen);
    RNSerialport.disconnect();
  }
  RNSerialport.stopUsbService();
};

onServiceStarted(response) {
  this.setState({servisStarted: true});
  if (response.deviceAttached) {
    this.onDeviceAttached();
  }
}
onServiceStopped() {
  this.setState({servisStarted: false});
}
onDeviceAttached() {
  this.setState({usbAttached: true});
}
onDeviceDetached() {
  this.setState({usbAttached: false});
}
onConnected() {
```

```
    this.setState({connected: true});
  }
  onDisconnected() {
    this.setState({connected: false});
  }
  onReadData(data) {
    if (
      this.state.returnedDataType === definitions.RETURNED_DATA_TYPES.INTARRAY
    ) {
      const payload = RNSerialport.intArrayToUtf16(data.payload);
      this.setState({output: this.state.output + payload});
    } else if (
      this.state.returnedDataType === definitions.RETURNED_DATA_TYPES.HEXSTRING
    ) {
      const payload = RNSerialport.hexToUtf16(data.payload);
      this.setState({output: this.state.output + payload});
    }
  }

  onError(error) {
    console.error(error);
  }

  handleConvertButton() {
    let data = '';
    if (
      this.state.returnedDataType === definitions.RETURNED_DATA_TYPES.HEXSTRING
    ) {
      data = RNSerialport.hexToUtf16(this.state.output);
    } else if (
      this.state.returnedDataType === definitions.RETURNED_DATA_TYPES.INTARRAY
    ) {
      data = RNSerialport.intArrayToUtf16(this.state.outputArray);
    } else {
      return;
    }
    this.setState({output: data});
  }
```

```
handleSendButton() {
  RNSerialport.writeString(this.state.sendText);
}

handleClearButton() {
  this.setState({output: ''});
  this.setState({outputArray: []});
}

buttonStyle = status => {
  return status
    ? styles.button
    : Object.assign({}, styles.button, {backgroundColor: '#C0C0C0'});
};

render() {
  return (
    <Container>
      {!this.state.usbAttached && (
        <>
          <SmartphoneSVG height={200} />
          <InsertDeviceWrapper>
            <UsbDeviceSVG width={100} height={100} />
            <ArrowUpSVG
              width={50}
              height={50}
              style={{position: 'absolute', left: 80}}
            />
          </InsertDeviceWrapper>

          <InstructionText>Insira o dispositivo</InstructionText>
        </>
      )}
      {this.state.usbAttached && (
        <>
          {/* <InstructionText>
            valor no aparelho:
          </InstructionText>
          <Input placeholder='insira o valor do aparelho'
```

```
            keyboardType='number-pad'/> */}
            <InstructionText>
              {this.state.output.length
                ? 'Resultado: '
                : 'Aguardando resultado...'}
            </InstructionText>
            <InstructionText>{this.state.output}</InstructionText>
          </>
        )}
      </Container>
    );
  }
}

const styles = StyleSheet.create({
  full: {
    flex: 1,
  },
  logo: {
    marginTop: 20,
    marginBottom: 20,
  },

  header: {
    display: 'flex',
    justifyContent: 'center',
    //alignItems: "center"
  },
  line: {
    display: 'flex',
    flexDirection: 'row',
  },
  line2: {
    display: 'flex',
    flexDirection: 'row',
    justifyContent: 'space-between',
  },
  title: {
    width: 100,
```

```
  },
  value: {
    marginLeft: 20,
  },
  output: {
    marginTop: 10,
    height: 300,
    padding: 10,
    backgroundColor: '#FFFFFF',
    borderWidth: 1,
  },
  inputContainer: {
    marginTop: 10,
    borderBottomWidth: 2,
  },
  textInput: {
    paddingLeft: 10,
    paddingRight: 10,
    height: 40,
  },
  button: {
    marginTop: 16,
    marginBottom: 16,
    paddingLeft: 15,
    paddingRight: 15,
    height: 40,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#147efb',
    borderRadius: 3,
  },
  buttonText: {
    color: '#FFFFFF',
  },
});

export default ManualConnection;
```