UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

PEDRO HENRIQUE PEREIRA

AMBIENTE PARA ANÁLISE DE ESPASTICIDADE EM LESADOS MEDULARES

São Carlos
2017

PEDRO HENRIQUE PEREIRA

AMBIENTE PARA ANÁLISE DE ESPASTICIDADE EM LESADOS MEDULARES

Monografia apresentada ao Curso de Engenharia Elétrica com ênfase em Eletrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Alberto Cliquet Júnior

São Carlos

2017

Pereira, Pedro Henrique
P372a Ambiente para análise de espasticidade em lesados medulares / Pedro Henrique Pereira; orientador Alberto Cliquet Júnior. São Carlos, 2017.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2017.

1. Espasticidade. 2. Lesão medular. 3. Computação em nuvem. 4. Sistemas embarcados. 5. RTOS. 6. IoT. I. Título.

FOLHA DE APROVAÇÃO

Nome: Pedro Henrique Pereira

Título: "Ambiente para análise de espasticidade em lesados medulares"

Trabalho de Conclusão de Curso defendido e aprovado em 23 / 11 / 2012,

com NOTA 10,0 (dez), **pela Comissão Julgadora:**

Prof. Titular Alberto Cliquet Júnior - Orientador - SEL/EESC/USP

Prof. Associado Evandro Luis Linhari Rodrigues - SEL/EESC/USP

Dr. Renato Varoto - Pós-doutorado/ UNICAMP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

DEDICATÓRIA

Aos meus pais.

“If you awaken from this illusion and you understand that black implies white, self implies other, life implies death (or shall I say death implies life?), you can feel yourself – not as a stranger in the world, not as something here unprobational, not as something that has arrived here by fluke - but you can begin to feel your own existence as absolutely fundamental.”

Alan Watts

RESUMO

A espasticidade é uma alteração motora caracterizada pela variação do tônus muscular identificada pelo aumento da resistência ao estiramento muscular. Este distúrbio aparece em consequência de diferentes patologias, destacando-se por uma maior frequência a lesão medular. A intensidade e a frequência da espasticidade em indivíduos com lesão medular podem gerar incapacidade, restringindo ou dificultando a realização de atividades cotidianas. Pelo fato de não existir cura definitiva até o momento, a qualidade de vida do indivíduo após a injúria está fortemente associada a qualidade e quantidade de intervenções fisioterapêuticas. Utilizando um sistema operacional em tempo real em microcontrolador de arquitetura ARM, um dispositivo de conectividade WiFi e uma aplicação em nuvem, neste trabalho é proposto um sistema completo para avaliação da espasticidade. Por meio do teste pendular é realizado amostragens de posicionamento angular com o uso de um eletrogoniômetro. Os dados são passados para a plataforma STM32F401 para processamento e análise, em seguida são transmitidos via serial para o dispositivo Nodemcu e por fim enviados para a nuvem via protocolo MQTT. A plataforma IBM Watson IoT é responsável por receber os dados e transmiti-los para uma aplicação em Node-RED na qual foi desenvolvido uma *dashboard* para visualização dos resultados obtidos pelo referido teste. O sistema, de modo geral, obteve um desempenho adequado, se portando como uma alternativa na validação da eficácia de determinado tratamento, uma vez que é capaz de identificar a evolução dos parâmetros do teste clínico. Para validação do ambiente proposto, este trabalho apresenta os resultados de dados previamente obtidos durante o tratamento da espasticidade por meio da estimulação elétrica neuromuscular e método *Kinesio Taping*.

Palavras chave: Espasticidade, Lesão Medular, Computação em nuvem, Sistemas embarcados, RTOS, IoT.

ABSTRACT

Spasticity is a motor disorder characterized by increased muscle tone identified by increased resistance to muscle stretching. This disorder appears as a consequence of different pathologies, among which the more frequent is the spinal cord injury. The intensity and frequency of spasticity in individuals with the injury can lead to disability, restricting or hampering daily activities. Because there is no definitive cure to date, the quality of life after spinal cord injury is strongly associated with the quality and quantity of physiotherapeutic interventions. Using a real-time operating system in ARM architecture microcontroller, a WiFi connectivity device and a cloud application, this work proposes a complete system for spasticity evaluation. Through the pendulum test, angular positioning samplings are made with the use of an electrogoniometer. The data is passed to the STM32F401 platform for processing and analysis, then transmitted via serial to the Nodemcu device which transfers to the cloud through MQTT protocol. The IBM Watson IoT platform is responsible for receiving the data and transmitting it to a Node-RED application in which a dashboard has been developed to visualize the results. The system has generally performed properly, behaving as an alternative in validating the efficacy of a given treatment, since it is able to identify the evolution of the parameters of the clinical test. To confirm the operation of the system, this work presents the results of data previously collected during the treatment of spasticity through neuromuscular electrical stimulation and Kinesio Taping method.

Keywords: Spasticity, Spinal Cord Injury, Cloud Computing, Embedded Systems, RTOS, IoT.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ADC – *Analog-to-Digital Converter*
API – *Application Program Interface*
ARM – *Advanced RISC Machine*
ASIA – *American Spinal Injury Association*
CoAP – *Constrained Application Protocol*
DDS – *Data-Distribution Service*
DSP – *Digital Signal Processor*
EENM – *Estimulação Elétrica NeuroMuscular*
ETSI – *European Telecommunications Standards Institute*
FPU – *Floating Point Unit*
GPIO – *General-purpose input/output*
GPL – *General Public License*
HAL– Hardware Abstraction Layer
I2C – *Inter-Integrated Circuit*
IaaS – Infrastructure as a service
IEEE – *Institute of Electrical and Electronics Engineers*
IoT – *Internet of Things*
ISR – *Interrupt service routine*
JSON – *JavaScript Object Notation*
KT – *Kinesio Taping*
M2M – *Machice-to-machine*
MAC – *Media access control*
MQTT – *Message Queue Telemetry Transport*
NoSQL DB – *Not only SQL(Structured Query Language) database*
PaaS – *Platform as a Servisse*
PWM – *Pulse Width Modulation*
RTOS – *Real-time operating ystem*
SaaS – *Software as a service*
SNC – Sistema Nervoso Central
SNP – Sistema Nervoso Periférico
SO – *Operating system*
SPI – *Serial Peripheral Interface*
SRAM – Static random-access memory

TCP – *Transmission Control Protocol*

UART – *Universal Asynchronous Receiver-Transmitter*

UDP – *User Datagram Protocol*

XMPP – *Extensible Messaging and Presence Protocol*

SUMÁRIO

1 – Introdução

A engenharia de reabilitação consiste na aplicação sistemática da engenharia e da ciências para o desenvolvimento, adaptação, aplicação e avaliação de soluções tecnológicas para problemas enfrentados por indivíduos com deficiência. Também tem como propósito a recuperação das funções físicas e cognitivas perdidas por motivos de doenças ou acidentes.

Nesta conjuntura, o desenvolvimento tecnológico tem contribuído para os progressos nas soluções de engenharia. A introdução da informática, computação em nuvem e o surgimento de aparelhos complexos trouxeram benefícios no tratamento e diagnóstico de doenças. Nota-se uma crescente busca tecnológica no setor médico, incentivada pela possibilidade de desenvolver sistemas que auxiliem a saúde da população. Em uma pesquisa realizada pela IDC, *International Data Corporation*, as indústrias que apresentarão o maior crescimento de receita em tecnologias cognitivas no período de 2016-2020 serão a *Healthcare* e Fabricação, uma taxa de crescimento anual composto de 69,3% e 61,4%, respectivamente.

Os avanços da microeletrônica e da ciência dos materiais permitiram desenvolver sensores e transdutores capazes de transformar um sinal físico em um sinal elétrico, permitindo assim coletar informações, responder a um estímulo e interagir com demais dispositivos. Atualmente, essa conectividade entre diversos dispositivos à internet é conhecida sob o conceito de *Internet of Things*.

No artigo *The Internet of Things for Health Care: A Comprehensive Survey,* Islam et al (2015) analisam os avanços nas tecnologias de sensoriamento médico e apresentam as arquiteturas e plataformas *state-of-the-art* em soluções baseadas em IoT. São apresentados os potencias da implementação da tecnologia como áreas de monitoramento de sinais fisiológicos e tratamento de doenças crônicas. Destacam-se também os principais casos de usos da tecnologia em ambiente médico: monitoramento de eletrocardiograma, registro de nível de glucose no sangue e pressão sanguínea, monitoramento de saturação de oxigênio e implementação de cadeiras de rodas inteligentes. Este conceito da aplicação de tecnologia em ambiente médico também é conhecido pelo termo *e-health*.

Dentro desse contexto de engenharia de reabilitação, uma aplicação de conectividade de sensores e *dashboard* de monitoramento se torna oportuna em exames que exigem uma medição de movimentos físicos, como ocorre no teste pendular realizado em indivíduos com lesão medular.

A lesão medular é caracterizada como toda injúria às estruturas contidas no canal medular (medula, cone medular e cauda equina), podendo levar a alterações motoras, sensitivas, autonômicas e psicoafetivas. (BRASIL, 2015). Estas alterações se manifestam em alguns distúrbios, dentre eles a paralisia ou paresia dos membros e alteração de tônus muscular, conhecido como espasticidade.

Avaliar o grau da espasticidade é fundamental para qualificar e quantificar o nível do distúrbio. Não existe uma intervenção para cura definitiva da lesão medular até o momento, então o tratamento da espasticidade baseia em minimizar a incapacidade do portador da lesão, seguindo um tratamento dentro de um programa de reabilitação. Dessa forma, são necessários ferramentais que registram e quantificam a evolução de determinado tratamento, e é função deste trabalho apresentar um sistema com essas características.

1.1 – Objetivos

Tem-se como principal objetivo deste trabalho o desenvolvimento de um sistema eficaz para avaliação de espasticidade em lesados medulares, auxiliando na realização do teste pendular, fornecendo indicadores por meio de um *dashboard* para auxílio da análise médica.

Como consequência, esse trabalho tem a proposta de desenvolver a integração entre os três níveis de arquitetura: sensor, hardware portando um sistema operacional em tempo real e a computação em nuvem.

1.2 – Justifivativa

Embora não existam dados precisos com relação à incidência de ocorrência da lesão medular no Brasil, estima-se que ocorram em torno de 6 a 8 mil novos casos por ano, sendo 80% homens e 60% com idade entre 10 e 30 anos (BRASIL, 2013).

A intensidade e a frequência da espasticidade em indivíduos com lesão medular podem gerar incapacidade, restringindo ou dificultando a realização de atividades diárias. Desta forma, a qualidade de vida após lesão medular está fortemente associada a qualidade e quantidade de intervenções fisioterapêuticas.

O tratamento da espasticidade deve ser sempre inserido em um programa de reabilitação e qualquer contribuição não invasiva que atenue a espasticidade, promova melhora da função motora e melhora da qualidade de vida do indivíduo acometido é preferível ao tratamento invasivo (KÓS, 2016). Por meio deste trabalho, será possível acompanhar o desenvolvimento de um ambiente de avaliação da espasticidade e efetividade de determinado tratamento, fornecendo indicadores para interpretação médica.

1.2 - Organização da monografia

Esta monografia está organizada em cinco capítulos. O primeiro capítulo apresenta uma introdução ao trabalho, seu objetivo e sua justificativa.

O segundo capítulo é responsável pela abordagem teórica, apresentando fundamentos biólogicos do tema, bem como os principais conceitos tecnológicos.

O terceiro capítulo descreve no sub-item materiais as plataformas e dispositivos utilizados no trabalho, e no sub-item métodos o desenvolvimento de cada ferramenta descrita na etapa anterior.

O capítulo quatro apresenta os resultados obtidos no decorrer do trabalho e uma discussão sobre os resultados finais.

O capítulo cinco apresenta as considerações finais, em forma de conclusão e considerações para trabalhos futuros.

2 - Fundamentos Teóricos

Para melhor compreensão deste trabalho é importante ter o conhecimento dos fundamentos biológicos das funcionalidades do sistema muscular, nervoso e motor, bem como a função da medula espinhal e as possíveis complicações que uma lesão nesse órgão pode ocasionar. Também é conveniente conhecer alguns conceitos tecnológicos que servem como sustentação no desenvolvimento de aplicações que integram dados e extração de informação.

2.1 - Fundamentos biológicos

2.1.1 - Sistema nervoso

O sistema nervoso pode ser divido em sistema nervoso central (SNC) e sistema nervoso periférico (SNP). A divisão é topográfica e também funcional (DANGELO e FATTINI, 2010).

O SNC é composto pelo encéfalo (cérebro, cerebelo e bulbo) e pela medula espinhal. O cérebro é o maior órgão do encéfalo e se encontra dentro da caixa craniana. Tem como função ser a matriz do sistema nervoso, servindo como controlador e regulador das atividades corporais, bem como receptor e interpretador dos estímulos sensoriais. Já as atividades do cerebelo estão relacionadas ao equilíbrio e postura corporal, também é responsável pelos reflexos e movimentos voluntários, agindo no sistema muscular como responsável pelo tônus musculares. O bulbo é o órgão que está em contato com a medula espinhal e serve como condução nervosa e centro de controle das funções vitais, como batimentos cardíacos, ritmo respiratório e pressão sanguínea.

O SNP corresponde as terminações nervosas, gânglios e nervos. As terminações nervosas existem nas extremidades de fibras sensitivas e motoras. Quando os receptores sensitivos são estimulados originam impulsos nervoso que caminham pelas fibras em direção ao SNC. Gânglio nervoso é o nome dado a acumulação de corpos celulares de neurônios encontrados fora do SNC, e apresentam-se, geralmente, de forma dilatada. Já os nervos podem ser classificados em nervos cranianos e nervos espinhais. São cordões esbranquiçados constituídos por fibras nervosas unidas por tecido conjuntivo e que tem como atividade levar ou trazer impulsos do SNC. Preliminarmente, deve-se ressaltar o fato de que as fibras de um nervo são classificadas de acordo com sua função. Por esta razão, diz-se que um nervo possui componente funcionais (DANGELO e FATTINI, 2010).

2.1.2 - Medula Espinhal

A medula espinhal é a porção mais caudal do SNC, localiza-se no canal raquidiano, iniciando logo abaixo do bulbo e estendendo-se até a primeira ou segunda vértebra lombar. É composta pelos segmentos cervicais, torácicos, lombares e sacrais. Os segmentos da medula cervical (C1 a C8) controlam a sensibilidade e o movimento da região cervical e dos membros superiores. Os segmentos torácicos (T1 a T12) controlam o tórax, abdome e parte dos membros superiores. Os segmentos lombares (L1 a L5) estão relacionados com movimentos e sensibilidade dos membros inferiores. Os sacrais (S1 a S5) controlam parte dos membros inferiores, sensibilidade da região genital e funcionamento da bexiga e intestino. A figura 1 ilustra essas divisões.

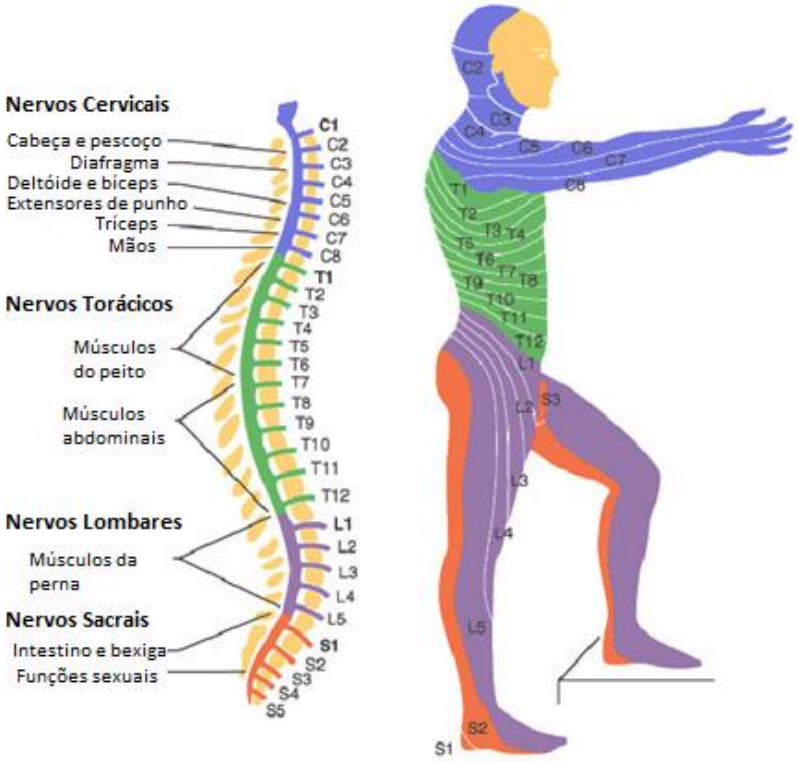

Figura 1 – Medula Espinhal – Divisões
Fonte: Adaptado de Next Step, Spinal Cord Injury Recovery. Disponível em: http://thenextstepsci.org.au/. Acesso em 02 Jun. 2017.

Cada segmento medular origina fibras nervosas ventrais que formam um par de raízes motoras e recebe um par de raízes dorsais, sensoriais. A união da raiz dorsal e da ventral constitui um nervo espinal de cada lado. Cada raiz sensorial é composta de fibras nervosas provenientes da pele, músculos, tendões, ossos e vísceras originadas do mesmo somito; a raiz motora que se une a ela é constituída por axônios que se destinam a músculos de mesma origem embriológica. A substância branca da medula espinal é constituída por conjuntos de

axônios ou fibras nervosas agrupadas, que recebem o nome de tratos, e situa-se ao redor da substância cinzenta. Na substância cinzenta, que em secção transversal tem forma de “H”, distinguem-se o corno anterior onde se situam os motoneurônios, o corno posterior cujos neurônios relacionam-se às vias sensoriais e a comissura cinzenta, onde se situa o canal central (NITRINI e BACHESCHI, 2003).

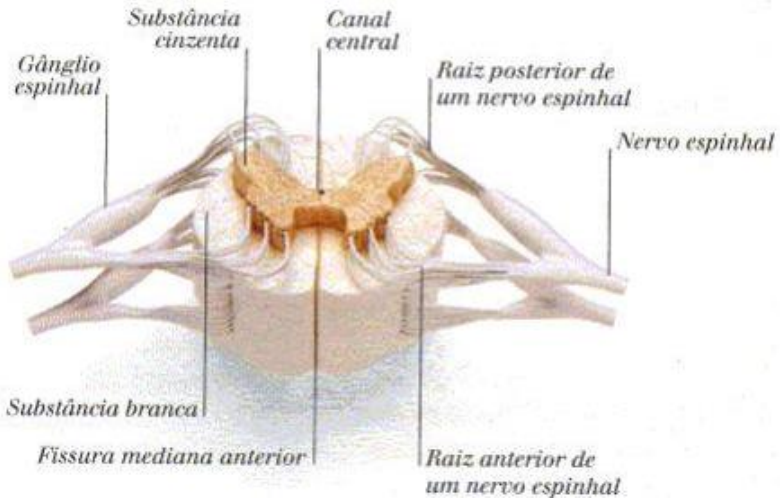

Figura 2 – Medula espinhal - secção transversal

Fonte: Adaptado de <http://www.infoescola.com/anatomia-humana/medula-espinhal/> Acesso em 05 Jun. 2017.

Alguns neurônios da medula espinhal participam de reflexos motores e neurovegetativos segmentares. O acionamento das unidades da coluna inter-mediolateral da medula espinal resulta na ativação das vias neurovegetativas simpáticas regionais, acarreta aumento da resistência vascular periférica e de vários órgãos, retenção urinária e alentecimento do trânsito intestinal. A ativação das unidades neuronais da ponta anterior da substância cinzenta da medula espinal causa hipertonia muscular que modifica o reflexo de flexão, gera aumento do tono e induz espasmos musculares, com a conseqüente redução da expansibilidade da caixa torácica que resulta em isquemia muscular, em anormalidades posturais e em síndrome dolorosa miofascial. A transferência das informações nociceptivas da medula espinal para estruturas encefálicas é realizada mediante vários sistemas de fibras longas representados pelo trato espinotalâmico, espinorre-ticular, espinomesencefálico, espinocervical, pós- sináptico do funículo posterior e trato intracornual. O maior contingente de tratos caudorrostrais envolvidos na nocicepção está presente no quadrante anterior da medula espinal (NITRINI e BACHESCHI, 2003).

Dentre as classificações existentes, o reflexo de estiramento (ou miotático) possui maior importância na compreensão da espasticidade. O reflexo miotático ou profundo é a contração brusca do músculo quando este é submetido a um estiramento rápido e tem por base o arco reflexo, situação ilustrada na figura 3.

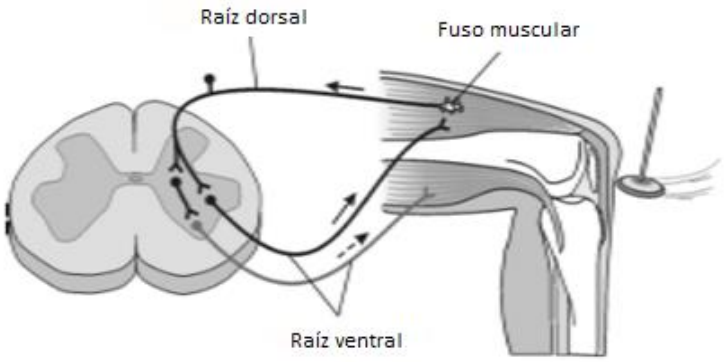

Figura 3 – Vias do arco reflexo
Fonte: Adaptado de NITRINI e BACHESCHI (2003).

2.1.3 - Lesão Medular

Chama-se de lesão medular toda injúria às estruturas contidas no canal medular (medula, cone medular e cauda equina), podendo levar a alterações motoras, sensitivas, autonômicas e psicoafetivas (BRASIL, 2015)

A lesão medular se caracteriza pela interrupção parcial ou total do sinal neurológico através da medula resultando em paralisia e ausência de sensibilidade do nível da lesão para baixo (KURTHY e ARAUJO, 2015). A medula pode ser lesada por acidentes, quedas, agressão por arma de fogo, acidentes automobilísticos e também ocasionada por doenças, como por exemplo, hemorragias, tumores e infecções por vírus.

Dois fatores influenciam no grau de limitação de cada indivíduo: O nível da lesão, isto é, a altura da lesão, sendo que uma injúria de nível mais alto resulta numa maior área corporal comprometida. E a extensão da lesão, se esta é completa ou incompleta. Dessa forma, podem existir tanto lesões altas incompletas como lesões baixas completas. (KURTHY e ARAUJO, 2015).

A lesão pode ser traumática ou não traumática e a classificação é realizada segundo a padronização internacional determinada pela ASIA – *American Spinal Injury Association,* em que força motora, sensibilidade e reflexos são examinados.

Considerando o nível da lesão medular, esta também pode ser categorizada em dois grupos: tetraplegia, que se refere a situação onde há grande comprometimento dos membros

superiores para baixo e a paraplegia, onde a função dos membros superiores é preservada, mas o tronco e os membros inferiores são comprometidos.

As alterações motoras e autonômicas pós lesão se manifestam principalmente como paralisia ou paresia dos membros, alteração dos reflexos superficiais e profundos, alteração ou perda das diferentes sensibilidades (tátil, dolorosa, de pressão, vibratória e proprioceptiva), perda de controle esfincteriano, disfunção sexual e alterações autonômicas como vasoplegia, alteração de sudorese, controle de temperatura corporal e alteração de tônus muscular, também conhecida como espasticidade.

2.1.4 Espasticidade

A espasticidade é uma alteração motora caracterizada por hipertonia e hiper-reflexia secundária a um aumento da resposta do reflexo de estiramento, diretamente proporcional a velocidade de estiramento muscular (SOCIEDADE PAULISTA DE MEDICINA FÍSICA E REABILITAÇAO, 2004). Este distúrbio aparece em diferentes doenças, dentre as quais destacam-se por maior frequência a paralisia cerebral, a lesão medular e a lesão encefálica, adquiridas por diferentes causas: traumáticas, tumorais, vasculares, infecciosas e degenerativas (MEYTHALER, 2001).

Em condições normais, os músculos apresentam certo grau de tono que pode ser examinado pela inspeção, palpação ou pela movimentação passiva. Na síndrome piramidal frequentemente há hiper-tonia porque a mobilização passiva estimula os fusos musculares e, através do arco reflexo, os motoneurônios. Como estes estão hiperativos, a contração reflexa é mais acentuada que em condições normais. A hipertonia que ocorre nas lesões dos neurônios motores superiores é denominada hipertonia espástica ou espasticidade. Como os fusos musculares são mais sensíveis ao estiramento rápido do músculo, a hipertonia é mais evidente quando os músculos são mobilizados com maior velocidade e, inversamente, menos evidente se os músculos forem mobilizados lentamente. Na espasticidade pode ser constatado o sinal do canivete. O estiramento passivo do músculo espástico encontra grande resistência inicial que cessa bruscamente, de modo semelhante ao que ocorre ao se abrir ou fechar um canivete. A redução brusca de resistência deve-se à estimulação de outro tipo de receptor contido nos fusos musculares e de outros mecanoceptores que provocam a inibição reflexa dos músculos submetidos ao estiramento (NITRINI e BACHESCHI, 2003)

A hiper-reflexia se caracteriza pelo aumento quantitativo da resposta do músculo estimulado, a diminuição do limiar de estimulação e o aumento da área reflexógena. (SOCIEDADE PAULISTA DE MEDICINA FÍSICA E REABILITAÇAO, 2004). A descarga repetitiva deste reflexo exacerbado origina o clonus. A lesão dos tratos piramidais na medula causa déficit motor, hiper-reflexia e espasticidade (NITRINI e BACHESCHI, 2003).

2.1.5 Avaliação e tratamento da espasticidade

Na avaliação da espasticidade são utilizados indicadores quantitativos e qualitativos. Estes são utilizados para identificar a intensidade e sua influência no desempenho da função, sendo úteis na indicação de intervenções terapêuticas e análise de seus resultados. (SOCIEDADE PAULISTA DE MEDICINA FÍSICA E REABILITAÇAO, 2004). Ao longo dos anos, foram elaboradas algumas formas de dimensionar os níveis de espasticidade. As principais delas são :

- Escala de Asworth e sua versão modificada: São as escalas mais utilizadas na avaliação do tônus muscular. Ashworth descreveu uma escala ordinal de 5 pontos para graduação da resistência encontrada durante o alongamento passivo, com 0 correspondendo a um tônus normal e 4 correspondendo a um aumento de tônus tão severo que a articulação se encontra rígida. Objetivando tornar a escala mais sensível a mudanças, BOHANNON e SMITH (1987) modificaram a escala de Ashworth acrescentado o grau " 1 +" e mudaram discretamente as definições, como apresentado na tabela 1.

Tabela 1 – Escala Modificada de Ashworth

Grau	Descrição
0	Nenhum aumento de tônus muscular
1	Leve aumento do tônus muscular, manifestado por uma tensão momentânea ou por resistência mínima, no final da amplitude de movimento articular (ADM), quando a região é movida em flexão ou extensão.
1+	Leve aumento do tônus muscular, manifestado por tensão abrupta, seguida de resistência mínima em menos da metade da ADM restante.
2	Aumento mais marcante do tônus muscular, durante a maior parte da ADM, mas a região é movida facilmente.
3	Considerável aumento do tônus muscular, o movimento passivo é difícil.
4	Parte afetada rígida em flexão ou extensão.

Fonte: SOCIEDADE PAULISTA DE MEDICINA FÍSICA E REABILITAÇAO (2004).

- Escala de avaliação de automatismo medulares: é uma escala ordinal que mede a frequência dos espasmos dos membros inferiores, de acordo com sua frequência por hora, escala de Penn apresentada na tabela 2, ou por comprometimento funcional, escala de Lyon Université, apresentada na tabela 3.

Tabela 2 – Escala de Penn, escore de frequência de espasmos.

Grau	Descrição
0	Ausente
1	Espasmos leves na estimulação
2	Espasmos infrequentes, menos de um por hora
3	Espasmos ocorrem, mais de um por hora
4	Espasmos ocorrem, mais de 10 vezes por dia

Fonte: SOCIEDADE PAULISTA DE MEDICINA FÍSICA E REABILITAÇAO (2004).

Tabela 3 – Escala de Lyon Université

Grau	Descrição
0	Ausência de automatismos.
1	Automatismos infrequentes ou de mínima intensidade, desencadeados por movimentos, não alteram postura nem função.
2	Automatismo frequentes ou de moderada intensidade, espontâneos, ou frente a movimentos, não prejudicam postura nem função.
3	Automatismo muito frequentes ou de grande intensidade que prejudicam postura e despertam a noite.
4	Automatismo constantes que impossibilitam a postura correta.

Fonte: SOCIEDADE PAULISTA DE MEDICINA FÍSICA E REABILITAÇÃO (2004).

A repercussão funcional da espasticidade nos indivíduos deambuladores pode ser analisada desde uma simples observação clinica até as formas mais detalhadas, como no laboratório de marcha, que auxilia a diferenciar alterações primarias e reações compensatórias. De todo modo, os parâmetros temporo-espaciais são os mais utilizados para avaliar o desempenho da marcha e a velocidade é a medida mais pratica de verificação (SOCIEDADE PAULISTA DE MEDICINA FÍSICA E REABILITAÇÃO, 2004)

2.1.6 - Teste Pendular

O teste pendular consiste em um método de avaliação do tônus muscular do quadríceps, analisando os efeitos causados pela espasticidade durante o balanço passivo do membro inferior. O procedimento é simples e não invasivo. Deve-se posicionar o paciente sentado sobre um apoio e deixar as pernas suspensas livremente. O examinador, então, levanta o membro inferior relaxado até a posição horizontal, deixando-a cair livremente, conforme ilustrado na figura 4.

Figura 4 – Teste pendular

Fonte: Adaptado de <http://onlinelibrary.wiley.com/doi/10.1111/j.1469-8749.2000.tb00067.x/epdf> Acesso em 15 Jul 2017.

Badje e Vodovnik em 1984 avaliaram o sinal do teste com o uso de um eletrogoniômetro caracterizando-o muito próximo de um sistema amortecido. Quantificaram as características do sinal em oito parâmetros. Os parâmetros dizem respeito aos números de oscilações antes da estabilização, ângulo máximo na primeira e segunda oscilação, tempo para entrar em regime estacionário e índice de relaxamento. A figura 5 ilustra um exemplo de sinal e o cálculo de três parâmetros.

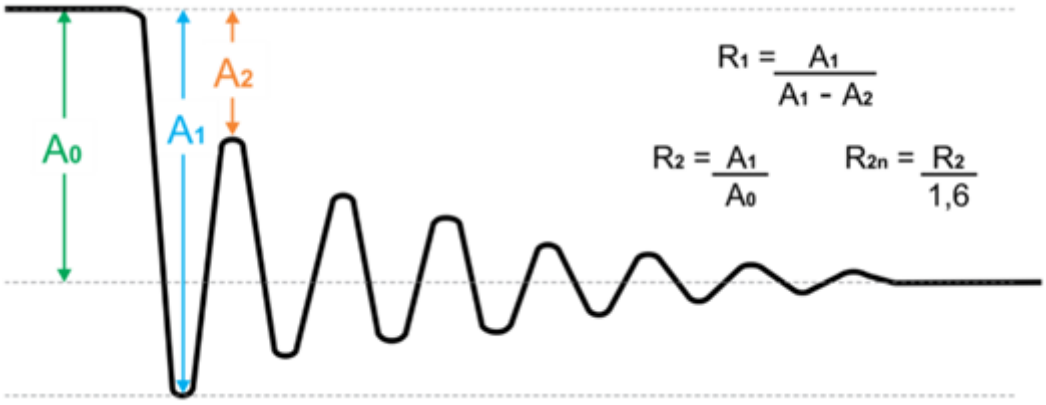

Figura 5 – Indicadores do teste pendular

Fonte: MARIA, 2015

Em indivíduos saudáveis, R_1 mostrou ser maior que 5 e em indivíduos com espasticidade mostrou ser aproximadamente 2.6. Já o parâmetro R_2 apresentou ser de 1.6 em indivíduos sem espasticidades. O parâmetro R_{2n} apresenta o indicador R_2 normalizado pelo

valor de 1.6 de indivíduos saudáveis, dessa forma R_{2n} >1 significaria um estado de não espasticidade e R_{2n} < 1 quantificaria o grau do distúrbio.

Com o sinal captado e os parâmetros calculados é possível analisar o estado de espasticidade do paciente, servindo como um método para quantificar a evolução e o efeito de um determinado tratamento.

2.1.7 Tratamento

Não existe uma intervenção para cura definitiva da lesão medular até o momento, então o tratamento da espasticidade baseia-se em minimizar a incapacidade do portador da lesão, seguindo um tratamento multifatorial dentro de um programa de reabilitação. O tempo de tratamento deve ser baseado na evolução funcional.

Dentre os métodos utilizados para o tratamento do referido distúrbio, encontram-se:

- Cinesioterapia: posturas e exercícios visando a introdução de padrões funcionais com objetivos de diminuição da hipertonia, fortalecimento da musculatura;
- Mecanoterapia: uso de equipamentos para realizar as atividades de cinesioterapia.
- Estimulação Elétrica Neuromuscular (EENM): permite a melhora da força de contração muscular, estimula a propriocepção, substitui o movimento e reduz a espasticidade (SOCIEDADE PAULISTA DE MEDICINA FISICA E REABILITAÇÃO, 2004);
- Medicamentos: drogas gabaergicas que agem por mecanismos que diminuem a excitabilidade dos reflexos medulares;
- Bandagem: recentemente, o método *Kinesio Taping* (KT) vem sendo utilizado por meio da estimulação cutânea para potencializar os estímulos somatosensoriais e têm demonstrando ajustes positivos acerca do tônus muscular (TAMBURELLA, 2014).

  Dentro do Método KT, a Técnica Muscular proporciona efeitos diretamente sobre a musculatura, sendo possível melhorar a contração de músculos ou grupos musculares enfraquecidos, hipotônicos e desequilibrados, além de diminuição de episódios de fadiga, contraturas, espasmos e lesões musculares (LEMOS, 2013).

  Um único estudo avaliou a influência do método KT comparado com bandagem placebo no controle do tônus muscular do tríceps sural em indivíduos com lesão medular incompleta de diversos níveis, observando possíveis efeitos na espasticidade, equilíbrio e marcha. O estudo notou

diferença significativa em todos os parâmetros avaliados, caracterizando o método KT como método válido para diminuição da espasticidade e melhora dos parâmetros relacionados à marcha em curto espaço de tempo (TAMBURELLA, 2014).

2.2 Conceitos tecnológicos

2.2.1 Internet das coisas

Internet das coisas, do inglês *Internet of Things*, ou simplesmente IoT, é um conceito que surgiu na década de 90 nos laboratórios do Massachusetts Institute of Technology – MIT a qual descrevia um sistema de sensores conectando o mundo físico a Internet.

Os avanços da microeletrônica e nanotecnologia transformaram o modo e velocidade da conectividade e que cada vez mais pequenos dispositivos possuem a capacidade de transformar uma variável física em dados elétricos e com isso coletar informações, conectando e interagindo com demais dispositivos.

O conceito de conectividade de dispositivos ultrapassou os limites de aplicações em automação que até então seria a área de maior aplicabilidade. O setor de agropecuária já conta com sensores espalhados em plantações coletando informações sobre temperatura, humidade do solo e probabilidade de chuva. O setor de logística também já utiliza sensores instalados em caminhões e contêineres para realizar o rastreio e coletar informações de trânsito para que a entrega seja realizada com segurança.

Similarmente, hospitais e clínicas representam uma das áreas de aplicação mais atraentes para IoT . IoT tem o potencial de dar origem a muitas aplicações médicas, tais como controle de saúde remoto, programas de ginástica, doenças crônicas e cuidados com idosos. A conformidade com o tratamento e a medicação em casa e pelos profissionais de saúde é outra aplicação potencial importante. Vários dispositivos médicos, sensores e dispositivos de diagnóstico e imagem podem ser vistos como dispositivos inteligentes ou objetos que constituem uma parte central da IoT. Os serviços de saúde baseados em IoT devem reduzir custos, aumentar a qualidade de vida e enriquecer a experiência do usuário. Do ponto de vista dos profissionais de saúde, IoT tem o potencial de reduzir o tempo de inatividade do dispositivo por meio da provisão remota (ISLAM et al, 2015).

Para que os dispositivos possam se comunicar é necessário que haja um protocolo. A escolha da tecnologia a ser empregada depende de alguns fatores, como o consumo de energia, a área de alcance, a largura de banda e a frequência a ser operada. O IEEE (*Institute of Electrical and Electronics Engineers*) e o ETSI (*European Telecommunications Standards*

Institute) listaram alguns dos protocolos de informação mais importantes, e entre eles estão o CoAP, MQTT, XMPP e DDS. E na camada de protocolo de transporte, as tecnologias mais relevantes são o WiFi, Bluetooth, ZigBee Smart, 3G/4G, LoRaWAN e Sigfox.

2.2.2 Computação em nuvem

Computação em nuvem se refere a uma tecnologia que permite acesso remoto a softwares, armazenamento de arquivos e processamento de dados por meio da internet, sendo uma alternativa a execução de um computador ou servidor local. Em outras palavras, é a entrega sob demanda de poder computacional por meio de uma plataforma com uma definição do preço conforme o uso.

Atualmente, pode-se categorizar a computação em nuvem em três grandes grupos relacionadas a serviços:

- SaaS – Software como serviço: caracteriza pela disponibilização de um software ao usuário por meio de uma interface de navegador ou de programa. O produto oferecido é executado e gerenciado pelo provedor de serviços.
- PaaS – Plataforma como serviço: o provedor é responsável pela manutenção do sistema operacional, da rede, dos servidores e da segurança. Pode-se também haver abstrações que aceleram o desenvolvimento de aplicativos.
- IaaS – Infraestrutura como serviço: proporciona as organizações a capacidade de aproveitar recursos brutos do servidor, enquanto o gerenciamento da plataforma e dos softwares é de responsabilidade da empresa. Dessa maneira, a empresa dispensa o investimento em hardware.

Quanto ao modelo de implementação, a computação em nuvem pode ser:

- Privada: nuvem construída exclusivamente para uma única organização.
- Pública: centenas de organizações podem usá-la de maneira simultânea. O provedor do serviço é responsável pelo gerenciamento e segurança.
- Hibrida: composição dos modelos de nuvens públicas e privadas. Permite que uma organização conecte a infraestrutura e aplicações entre recursos da web e recursos atuais que não se encontram na nuvem.

O uso da computação em nuvem traz uma série de vantagens para organizações. A principais são: flexibilidade, a qual os serviços em nuvem podem atender sob demanda, atualizações automáticas de software, diminuição de manutenção da infraestrutura física, eliminação de custo com alocação ociosa em datacenter, agilidade no processo de desenvolvimento de aplicações, e acesso em qualquer lugar.

Como ponto negativo, a computação em nuvem depende intrinsicamente da banda de internet e sua intermitência. Com isso, a velocidade de troca de informação pode ser afetada. Outro ponto de atenção é a questão da segurança. A manipulação de dados sensíveis, apesar de serem criptografados, ainda sofre a possibilidade de ataques cibernéticos, causando uma relativa resistência nas organizações em deixar os dados na nuvem.

2.2.3 Computação cognitiva

O termo computação cognitiva se refere a uma competência tecnológica inspirada nas capacidades do cérebro humano de analisar e resolver problemas. Com base no uso de sensores, modelos, algoritmos e dados, são desenvolvidos sistemas capazes de identificar padrões, fazer análise preditiva, validar hipóteses, elaborar análise probabilística com objetivo de ajudar o ser humano a tomar decisões adequadas e com mais assertividade. Em outras palavras, a computação cognitiva nasceu da interação do estudo do cérebro humano e da ciência da computação.

A base da computação cognitiva está no processamento de dados estruturados e não estruturados. E para isso conta com um conjunto de modelos baseados em inteligência artificial. Inicialmente é necessário selecionar o domínio (assunto) e o conteúdo apropriado dentro daquele domínio, também conhecido como *corpus.* Para gerar o conteúdo apropriado, é necessário envolver os especialistas sobre o domínio que se deseja ensinar a um sistema cognitivo. A razão pela qual o ser humano tem um papel importante nesse processo é porque sistemas cognitivos se tornam inteligentes com o tempo e com dados relevantes (ROTTA e VARGA, 2016).

Com os dados coletados e tratados, é necessário treinar o sistema para ocorrer o aprendizado. O processo de aprendizado pode contar com algoritmos de *machine learning*, *deep learning*, sistemas baseados em regras, redes neurais e sistemas probabilístico.

Com esse conjunto, os sistemas cognitivos conseguem assimilar de forma estatística o motivo de um determinado dado ser mais ou menos relevante, utilizando seu *corpus* como base de evidências e melhorando com o tempo por meio do feedback positivo ou negativo durante a utilização do sistema (ROTTA e VARGA, 2016).

Um exemplo do uso de sistemas cognitivos está na área da saúde. Um sistema pode reter milhares de informações, processa-las e tirar *insights* relevantes, como é o caso do *Watson Oncology*. Trata-se de um sistema que conta com um banco de dados de milhares de pesquisas científicas e históricos de pacientes, gerando uma recomendação de tratamento de câncer e fornecendo indicadores para realização de um diagnóstico mais preciso.

Reconhecimento de fala, traduções, conversão de voz para texto, análises de sentimentos e reconhecimento de objetos em imagens também são exemplos de sistemas que utilizam o conceito de computação cognitiva.

3 - Materiais e Métodos

3.1 Desenvolvimento

Para o desenvolvimento de um ambiente de monitoramento de sinais, propósito deste trabalho, se faz necessário a integração de sensores, plataforma de hardware e plataformas de software. A figura 6 apresenta a arquitetura de integração funcional do projeto. Cada componente presente na imagem será introduzido no item materiais e os procedimentos de desenvolvimento, integração e execução final serão descritos no item métodos.

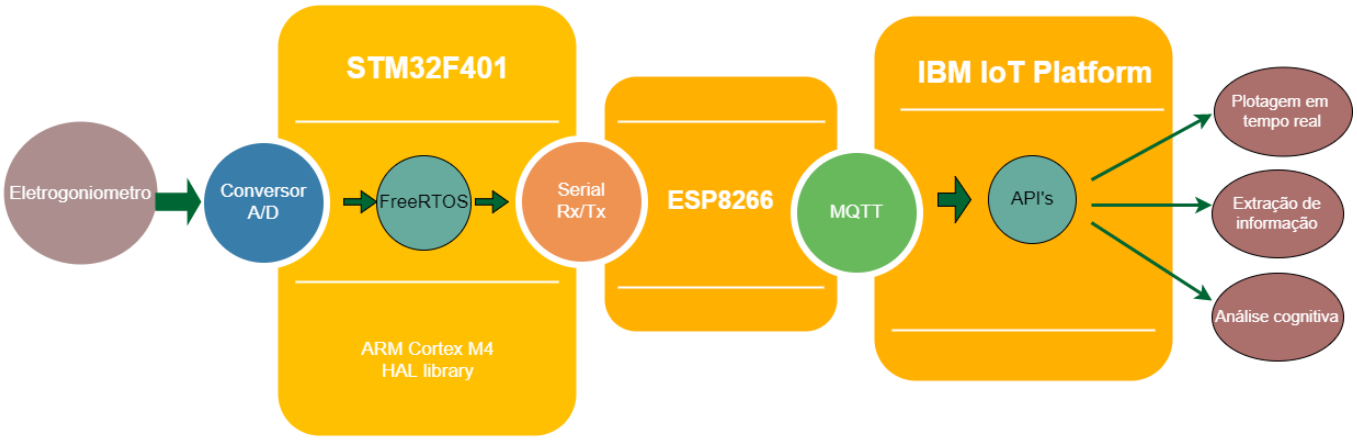

Figura 6 - Arquitetura de integração

Fonte: Autoria própria

3.2 Materiais

3.2.1 Eletrogoniômetro

O goniômetro consiste de um aparelho para medição de ângulos, podendo ser de contato ou de reflexão. Os goniômetros de contato são aplicados sobre o corpo que se pretende medir, aferindo o ângulo mecanicamente. Já os de reflexão utilizam propriedades ópticas, como o uso por fibra óptica, caso do eletrogoniômetro.

O eletrogoniômetro é um instrumento não invasivo que registra o deslocamento angular baseado no comportamento da fibra óptica. A fibra óptica é composta pelo núcleo, casca e revestimento primário, podendo conter outras malhas externas de proteção mecânica. A figura 7 ilustra sua estrutura básica.

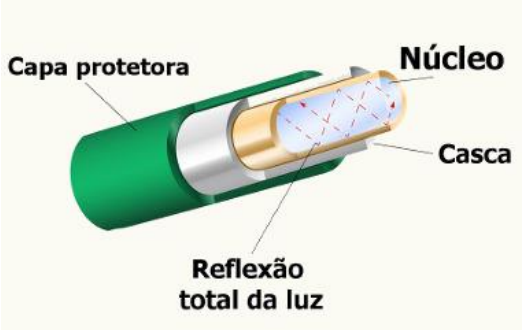

Figura 7 – Estrutura básica de uma fibra óptica.

Fonte: Adaptado de < http://alunosonline.uol.com.br/fisica/fibras-opticas.html > Acesso 20 Jul 2017.

O núcleo possui índice de refração maior que a casca. Essa diferença é necessária para a condição de confinamento e propagação da luz.

O eletrogoniômetro de fibra ótica utiliza o princípio de modulação de intensidade. Através da fibra, ocorre-se uma atenuação do sinal luminoso. Essa perda de potência é devido a:

- Absorção: parte do sinal é absorvido pelo material por fator intrínsecos e extrínsecos.
- Espalhamento: reduções na amplitude do campo guiado por mudanças na direção de propagação, causadas pelo próprio material e por imperfeições no núcleo da fibra.
- Curvatura: macrocurvaturas: a ocorrência da perda é dada quando os modos próximos ao ângulo crítico (alta ordem) ultrapassam este valor em função da curvatura. Assim deixam de ser totalmente refletidos internamente, passando a ser refratados; Microscópicas: resultam de flutuações aleatórias de pequena escala na fronteira entre núcleo e a casca.
- Projeto de guias de ondas: perdas em conectores, emendas, desalinhamento axial e rugosidade nas extremidades.

No caso do eletrogoniômetro, o fator de curvatura é o maior responsável pela atenuação na fibra, permitindo assim modular a diferença da luz emitida e recebida relacionando com a curvatura realizada pela fibra. A figura 8 apresenta um eletrogoniômetro em curvatura.

Figura 8 – Eletrogoniômetro em curvatura.
Fonte: Adaptado de < http://www.alliantech.com/pdf/capteurs_mouvements/S700.pdf> Acesso em 20 Jul. 2017.

O sensor Shape sensor s700 joint angle utilizado para mensurar os dados de paciente neste trabalho relaciona linearmente os movimentos angulares com a tensão de saída, de acordo com o fabricante. O sensor realiza o condicionamento do sinal e permite uma calibragem de offset. As demais características informadas pelo *datasheet* são:

- Faixa de escala: ± 1V para ± 90º
- Tensão de saída para ângulo de 90º: 2,5V ± 0,2V
- Resolução: 0,05
- Frequência de corte: 1kHz
- Temperatura: 0-50 ºC
- Tensão de alimentação: 5-15V
- Peso: 45g

3.2.2 Plataforma STM32F401

A plataforma STM32F401 é um sistema embarcado que pertence à família Nucleo Board de microcontroladores de alta performance e baixo consumo produzido pela empresa franco-italiana STMicroelectronics. É composto por um microcontrolador de arquitetura ARM 32 bits, Cortex-M4. A plataforma conta com uma Unidade de Ponto Flutuante (FPU, *Floating Point Unit*) e instruções de DSP (Digital Signal Processing), muito útil quando se necessita de manipulação matemática, processamento e filtragem de dados digitais em ambiente embarcado, sendo os principais motivos que levaram à escolha da plataforma, quando comparada a microcontroladores populares de 8 bits, como PICs e AVRs. Em quesito de

conectividade física, a plataforma possui *headeres* padrão Arduino e Raspberry PI para rápida prototipagem e compatibilidade de *shields* das plataformas populares. A figura 9 apresenta a plataforma.

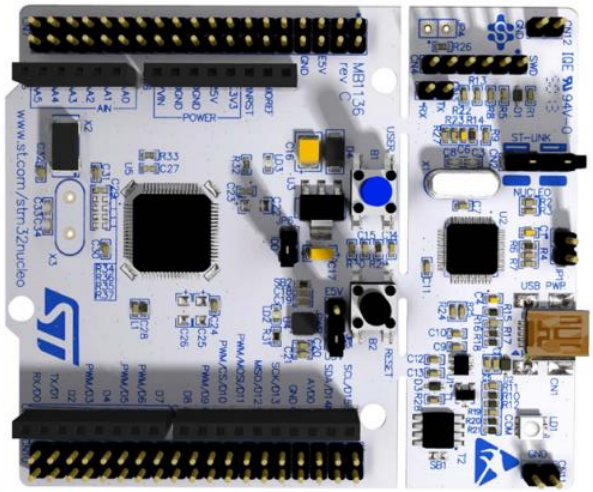

Figura 9 - STM32F401RE Nucleo Board.

Fonte: Adaptado de <https://www.element14.com/community/docs/DOC-69701/l/stm32-nucleo-development-board-for-stm32-f4-series-with-stm32f401re-mcu >. Acesso em10 Mai. 2017.

O modelo STM32F401RE, utilizado neste trabalho, opera numa frequência de até 84MHz, possui 512 Kbytes de memória *flash* e 96Kbytes de memória SRAM. Outras características da plataforma são:

- 3x USART a 10.5Mbit/s
- 81 GPIO
- 4x SPIS a 42Mbit/s
- 3x I²C
- 1x SDIO
- 1x USB OTG full speed
- 2x full duplex I²S até 32-BIT/192KHz,
- Serial wire debug (SWD) e JTAG modos de debug
- 1x 12-bit ADC, até 16 canais
- 10 timers, 16 e 32 bits, até 84MHz

3.2.3 Sistema operacional em tempo real

Em sistemas embarcados, pode-se programar uma aplicação em dois grandes métodos. O primeiro método trata-se do *bare-metal*. É o modo onde o controle de hardware, uso de memória, tarefas e interrupções são gerenciadas no próprio código do desenvolvedor. É um método muito comum para microcontroladores de 8 e 16 bits utilizando linguagem C e/ou

Assembly, necessitando amplo domínio de baixo nível de hardware, permitindo alta performance da aplicação.

Um outro método, caracterizado por uma abstração do hardware, é a utilização de um Sistema Operacional. Pode-se utilizar um sistema operacional completo, como as distribuições Linux e Windows CE ou um sistema operacional em tempo real mais enxuto, menos abstrato e otimizado para aplicações especificas. Este último método foi selecionado para o desenvolvimento deste trabalho pela confiabilidade da execução de tarefas em um tempo determinístico, compatibilidade da plataforma de hardware escolhida e outras vantagens e características expostas a seguir.

3.2.4 FreeRTOS

O sistema operacional em tempo real selecionado para a aplicação foi o FreeRTOS, por se tratar de um software open source sob licença GPL bem consolidado e líder de mercado. É mantido pela empresa Real Time Engineers e foi desenvolvido para ser simples e leve. Seu *kernel* possui cinco arquivos essenciais escritos em linguagem C, que tratam sobre os temporizadores, tarefas, rotinas, mensagens, filas e listas. A árvore de arquivos do *kernel* está ilustrada na figura 10.

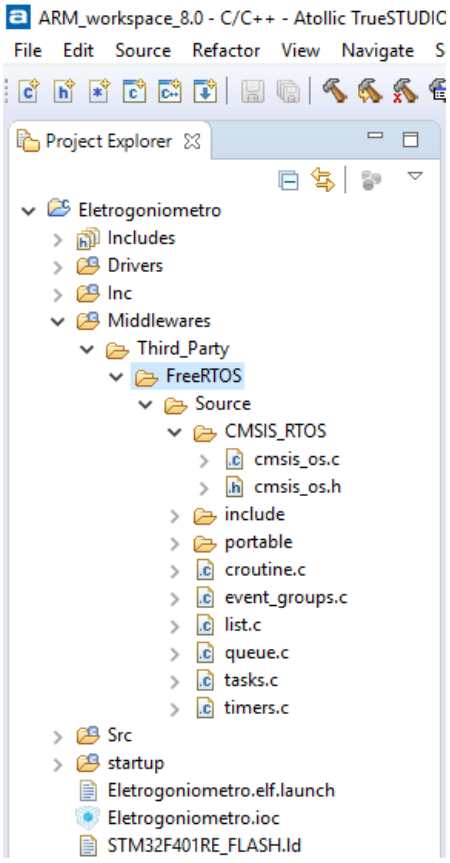

Figura 10 – Principais arquivos do FreeRTOS.

Fonte: Autoria própria

Um RTOS é destinado a execução de múltiplas tarefas (comumente chamadas de *multi tasks* ou *multi threads*) onde o tempo de resposta a um evento é pré-definido. Cada tarefa possui um nível de prioridade, desta forma, o *kernel* necessita de algum mecanismo de gerenciamento, classificando as tarefas e ordenando-as de acordo com suas prioridades. Esse procedimento é feito por meio do escalonamento de tarefas. Um exemplo de escalonamento de tarefas está apresentado na figura 11.

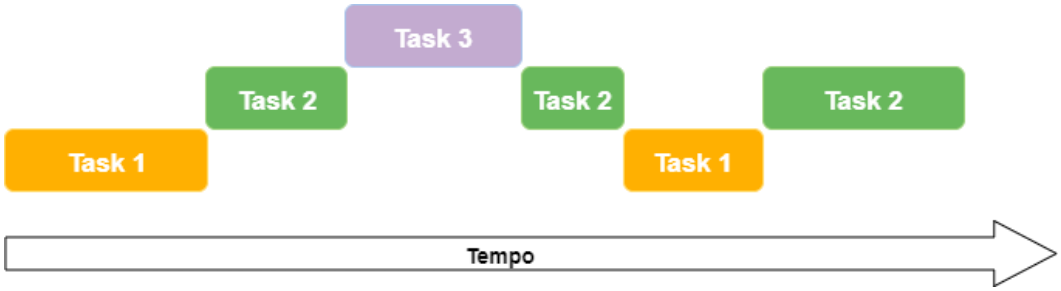

Figura 11 – Escalonamento na execução de tarefas.

Fonte: Adaptado de <http://www.freertos.org/implementation/a00005.html>. Acesso em 15 Abr. 2017.

Por esse motivo, o escalonador do SO tem papel fundamental em garantir que uma tarefa foi iniciada e finalizada no tempo definido, para assim dar o *start* na tarefa seguinte que já se encontrava em *status* de espera. A figura 12 apresenta o comportamento de uma *task* no FreeRTOS.

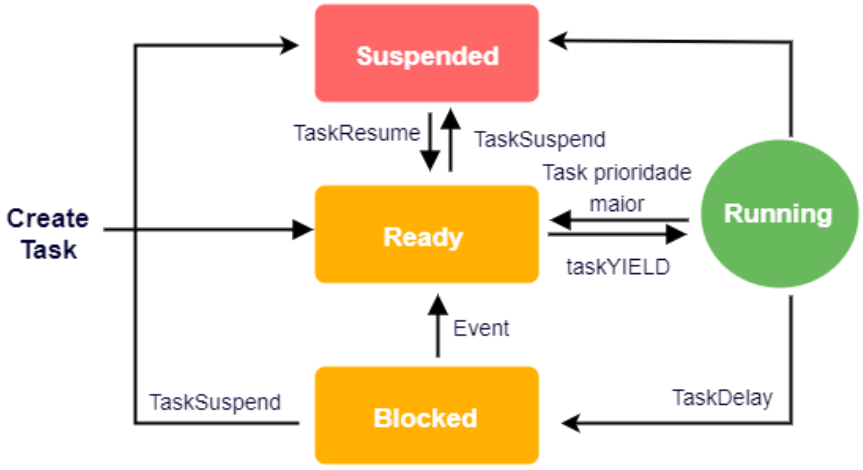

Figura 12 – Estados das tarefas.

Fonte: Adaptado de <http://www.emcu.it/STM32F4xx/Exe2_FreeRTOS_on_STM32F4-Discovery/TASKRunning-Ready-Blocked-Suspended.png>. Acesso em 15 Abr 2017.

A comunicação entre tarefas, como envio e recebimento de mensagens, pode ser efetuada pelas filas (*Message Queues*) e pelos semáforos (*Semaphores*), que são mecanismo para o compartilhamento de recursos.

Dessa forma, um RTOS provê toda esta arquitetura de software para o desenvolvimento de Sistemas de Tempo Real, incluindo funções para a criação de tarefas, troca de mensagens entre as tarefas, compartilhamento de recursos e ISR's genéricas (PRADO, 2010).

3.2.4 Módulo Wi-Fi - ESP8266-Nodemcu

O ESP8266 é um microcontrolador capaz de fazer comunicação Wi-Fi. O dispositivo é originalmente fabricado pela empresa chinesa Espressif e se difundiu rapidamente nos projetos de IoT devido ao seu baixo custo.

WiFi é um outro nome dado ao protocolo IEEE 802.11, atua na camada física, e define padrões de transmissão e codificação. O modulo permite se conectar a uma rede sem fio fazendo conexões TCP/IP.

O modelo utilizado neste projeto é o Nodemcu ESP-12 por já possuir um conversor USB – serial (CH340) e um regulador de tensão de 3,3V. Suporta redes 802.11 b/g/n podendo trabalhar em modo de ponto de acesso (*Acess Point*) ou como estação (*Station*) enviando e recebendo dados.

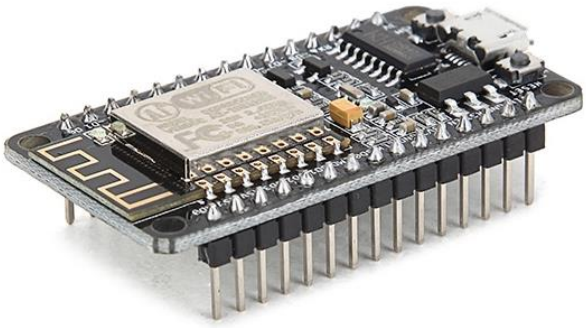

Figura 13 – Modelo NodeMcu ESP-12.

Fonte – Adaptado de < https://www.filipeflop.com/produto/modulo-wifi-esp8266-nodemcu-esp-12/> Acesso em 18 Apr. 2017.

Outras características do modelo ESP-12 são

- Tensão de operação: 3,3V;
- Alcance: 90m aprox.
- Faixa de frequência: 2.4GHz
- Comunicação: Serial (TX/RX)
- Suporta comunicação TCP e UDP

- Conectores: GPIO, I2C, SPI, UART, Entrada ADC, Saída PWM e Sensor de Temperatura interno.
- Modo de segurança: PEN/WEP/WPA_PSK/WPA2_PSK/WPA_WPA2_PSK
- Dimensões: 25 x 14 x 1mm
- Peso: 7g

A configuração dos 8 pinos disponíveis está presenta na figura 14.

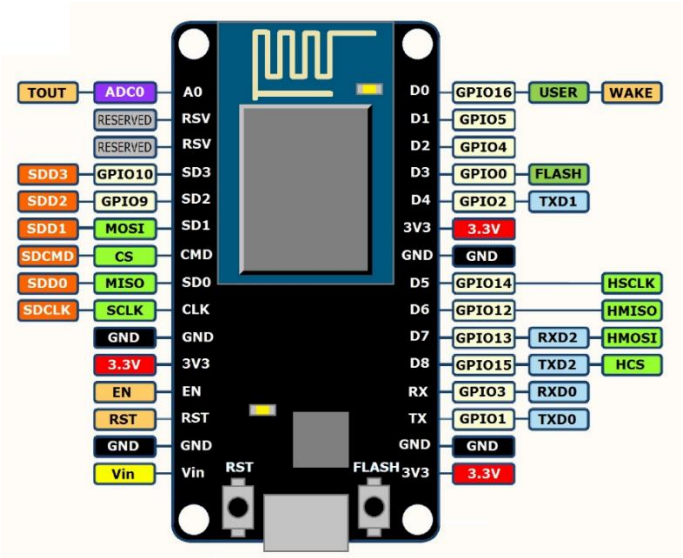

Figura 14 – Configuração dos pinos do modelo NodeMcu ESP-12.
Fonte: Adaptado de <http://crufti.com/getting-started-with-espruino-on-esp8266/>. Acesso em 22 Abr. 2017.

Na arquitetura do projeto apresentado na figura 6 da seção 2.6, o módulo Nodemcu funciona como estação, recebendo os dados via interface serial (Tx/Rx) da plataforma STM32F401 e enviando via protocolo MQTT para a nuvem.

3.2.5 MQTT

O MQTT, do inglês *Message Queue Telemetry Transport*, é um protocolo de conectividade *machice-to-machine* (M2M) desenvolvido pela IBM em 1999 com objetivo de ser leve, simples e de mínima utilização da largura de banda. É baseado no padrão de troca *publish/subscribe*.

Nesse padrão, quando um dispositivo deseja receber um dado, ele subscreve, fazendo uma requisição para um elemento intermediário chamado de *Broker*. Da mesma forma, elementos da rede que desejam publicar informações, o fazem também pelo intermédio do Broker. Assim, o elemento intermediário e centralizador tem a função de gerenciar as publicações e subscrições. A figura 15 apresenta um exemplo dessa configuração.

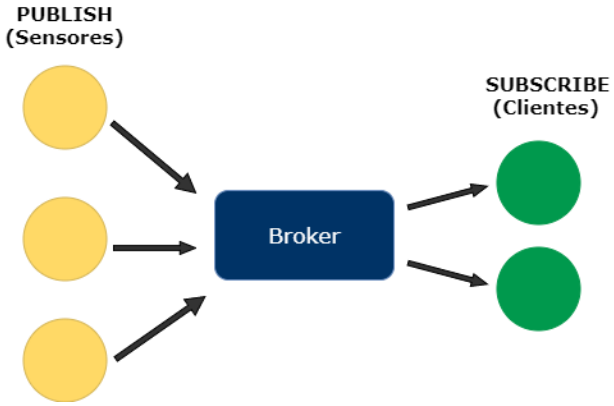

Figura 15 – Exemplo de configuração do Broker.

Fonte: Autoria própria

Atualmente esse protocolo é amplamente utilizado em redes de conceito *Internet of Things*.

3.2.6 Plataforma IBM Cloud Bluemix

IBM Cloud Bluemix é uma Plataforma Como um Serviço (Platform as a Servisse - PaaS) que permite ao desenvolvedor criar, implementar e gerenciar aplicações na nuvem de maneira rápida e segura. O Bluemix foi escolhido como plataforma neste trabalho por possuir uma integração completa de serviço de IoT, banco de dados e NodeRED,. A plataforma também conta com mais de 150 serviços, incluindo Watson APIs, Blockchain, OpenWhisk, plataformas de *analytics* e diferentes frameworks.

Baseado no conceito de computação cognitiva, a plataforma fornece a capacidade de criar uma aplicação capaz de entender dados, aprender com eles e raciocinar a partir deles. A figura 16 apresenta algumas APIs disponíveis na plataforma.

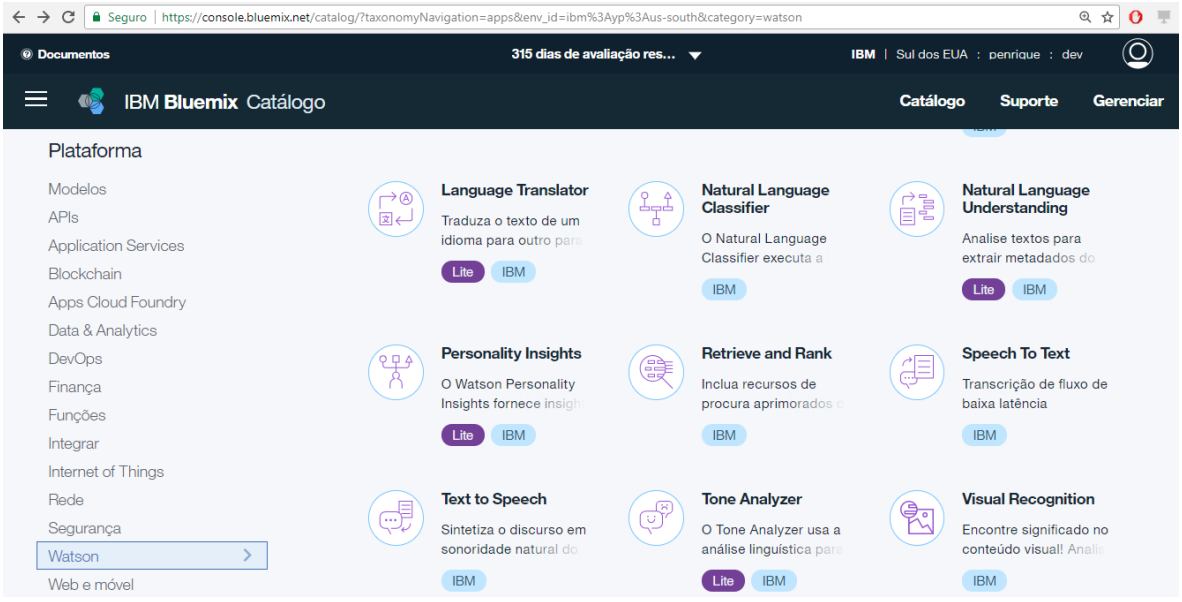

Figura 16 – IBM Cloud Bluemix – Watson APIs.

Fonte - Autoria própria

Na realização deste trabalho, utilizou-se os serviços de IoT, apps Cloud Foundry e Banco de Dados. A configuração da aplicação será descrita em um dos itens da próxima etapa.

3.3 Métodos

3.3.1 Configuração do STM32F401

As configurações dos periféricos da plataforma stm32F401 foram realizadas por meio da ferramenta STM32CubeMX utilizando o *HAL driver* disponibilizado pela STMicroelectronics. Trata-se de uma camada de driver que prove APIs para interação entre camadas de aplicação, hardware e demais bibliotecas. É conveniente lembrar que todas as tarefas do código ficam sob regência do sistema operacional em tempo real freeRTOS. O processo de codificação, implementação, testes e debug foi realizado pela IDE Atollic TrueStudio, ambiente versátil para desenvolver aplicações em microcontroladores ARM.

O arquivo *main.c*, apêndice A, tem como primeira função ler constantemente da entrada analógica e converter para digital, com 8 bits de resolução e taxa de amostragem de 200 *samples/s*, passível de calibração. Em seguida é realizado a conversão entre o sinal de tensão lido e o ângulo correspondente do eletrogoniômetro. Como se trata de uma operação linear a conversão é feita através da expressão: $f(x) = \frac{180xV}{3.3}$, considerando V como o valor do tensão lido pelo ADC e 3,3V o fundo de escala de referência.

O valor já convertido em ângulo é então enviado via serial para o Nodemcu e armazenado em um buffer. O procedimento é repetido até que se finaliza o período de leitura, que também é configurável. Com todos os dados presentes no buffer, é então rodado um algoritmo para identificar o primeiro ponto de pico, o primeiro vale e o valor em regime estacionário. O princípio de uma interação do algoritmo está descrito no quadro abaixo.

$$Se\ \left(buf[n] > \ (buf[n+1] + \ tol)\ \ e\ \ buf[n] > \ (buf[n-1] + \ tol)\right)$$

$$Então\ A_1 = buf[n]$$

$$Se\ \left(buf[n] < \ (buf[n+1] - \ tol)\ \ e\ \ buf[n] < (buf[n-1] - \ tol)\right)$$

$$Então\ A_2 \ = \ buf[n]$$

$$Se\ (buf[n] - \ buf[n \ + \ m] < \ tol2\ )$$

$$Então\ A_0 = \frac{1}{m+1} \sum_{i=n}^{i=n+m} buf[i]$$

$$n = n + 1$$

Considerando:

- **tol** e **tol2** representam uma possível tolerância devido a presença de flutuações do sinal no movimento no teste pendular, como por exemplo na sequência de leitura: [...40.7, 41.2, 40.8, 41.1, 41.2, 40.8...]
- **m** representa a extensão para analisar quando o sistema se encontra em regime estacionário.
- **n** representa o índice do *buffer.*

Finalizado a identificação dos indicadores relevantes, uma flag é enviada para o Nodemcu sinalizando que os próximos valores enviados serão R1, R2 e R2n, estritamente nessa ordem.

Pode-se encontrar todo o código desta parte do projeto por meio do seguinte link do github https://github.com/pedrohenriqp/rtos_espasticidade.

3.3.2 Configuração do Nodemcu

Para efetuar a comunicação entre o módulo WiFi e a nuvem, utilizou-se a biblioteca PubSubClient que suporta o protocolo MQTT pelo método publish/subscribe. É compatível com a IDE do Arduino e está disponível em https://github.com/knolleary/pubsubclient . O código presente no apêndice B apresenta as instruções para identificar o endereço MAC do dispositivo. Esse dado é necessário para cadastrar o ESP8266 como um dispositivo na aplicação do Bluemix.

Para criação de um serviço IoT, basta acessar o catálogo do Bluemix e selecionar o ícone de Plataforma de Internet das Coisas. Logo após a criação, foi realizado a inclusão e o registro de dispositivo. As figuras 17 e 18 apresentam o procedimento.

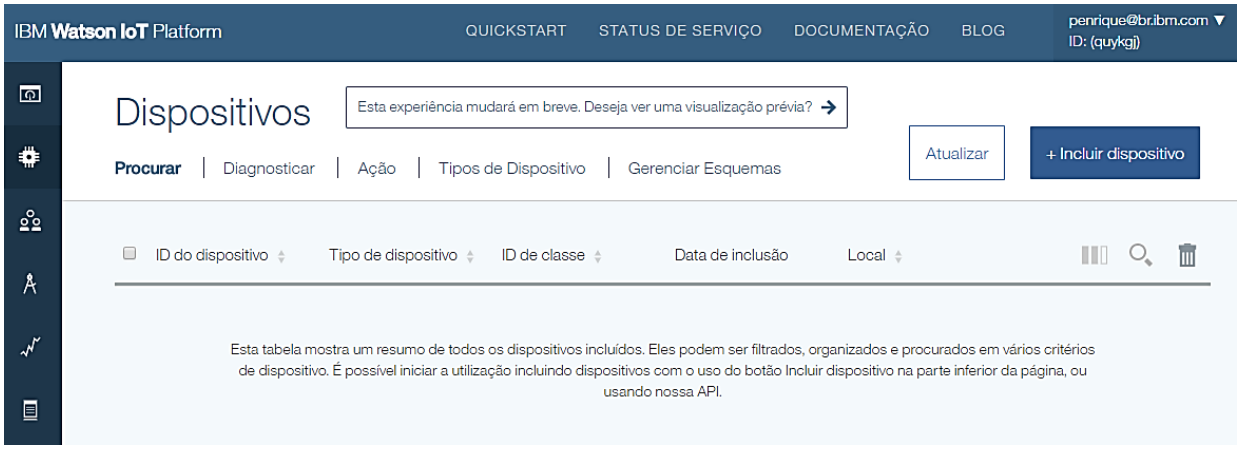

Figura 17 – Área de controle da IBM Watson IoT Platform.

Fonte: Autoria própria.

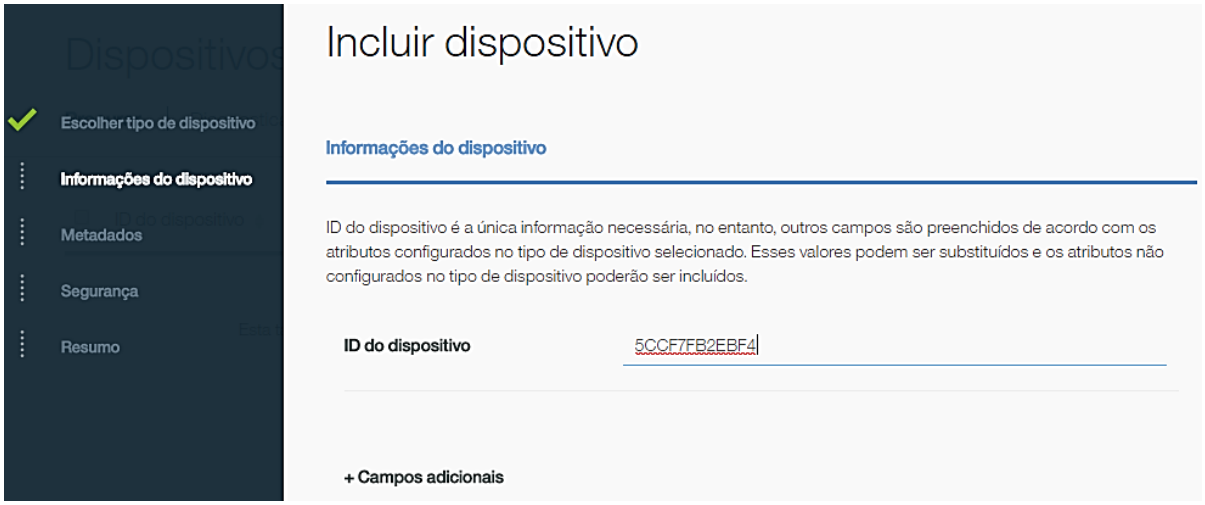

Figura 18 – Inclusão e registro de dispositivo por meio do endereço MAC.

Fonte: Autoria própria.

Após registro e inserção de outros dados para identificação, é gerado um Token de autenticação. Esse Token é indispensável na aplicação pois é ele que certifica cada sensor em uma rede de dispositivos. A figura 19 apresenta os dados relevantes de identificação.

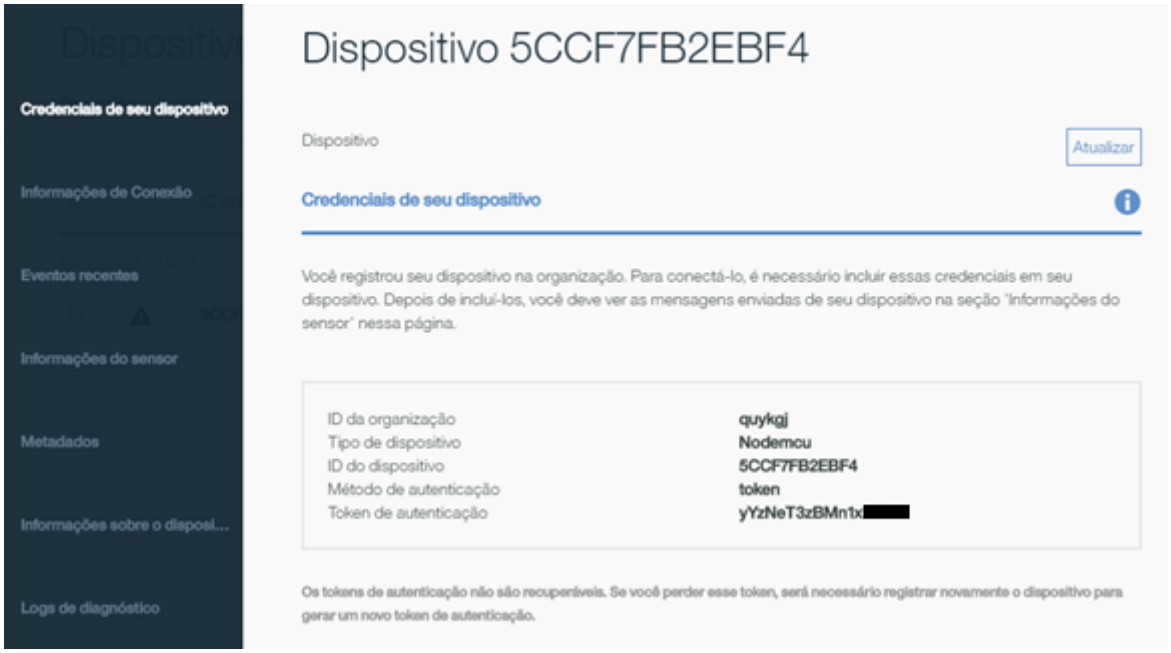

Figura 19 – Credenciais do dispositivo.

Fonte: Autoria própria.

Com as credenciais do dispositivo, foi desenvolvido um *script* no Nodemcu para enviar um *payload* em formato JSON para testar a conectividade, com objetivo de avaliar o registro do evento e a integridade da informação. A figura 20 apresenta o sucesso do teste.

Figura 20 – Registro de evento.

Fonte : Autoria própria.

Já na aba de *Cards* da área de controle, é possível adicionar diferentes gráficos e telas de informações sobre o dispositivo conectado. A figura 21 apresenta um exemplo de um teste de conectividade. Nela são exibidos dados recebidos do Nodemcu por MQTT, podendo ser plotados em tempo real.

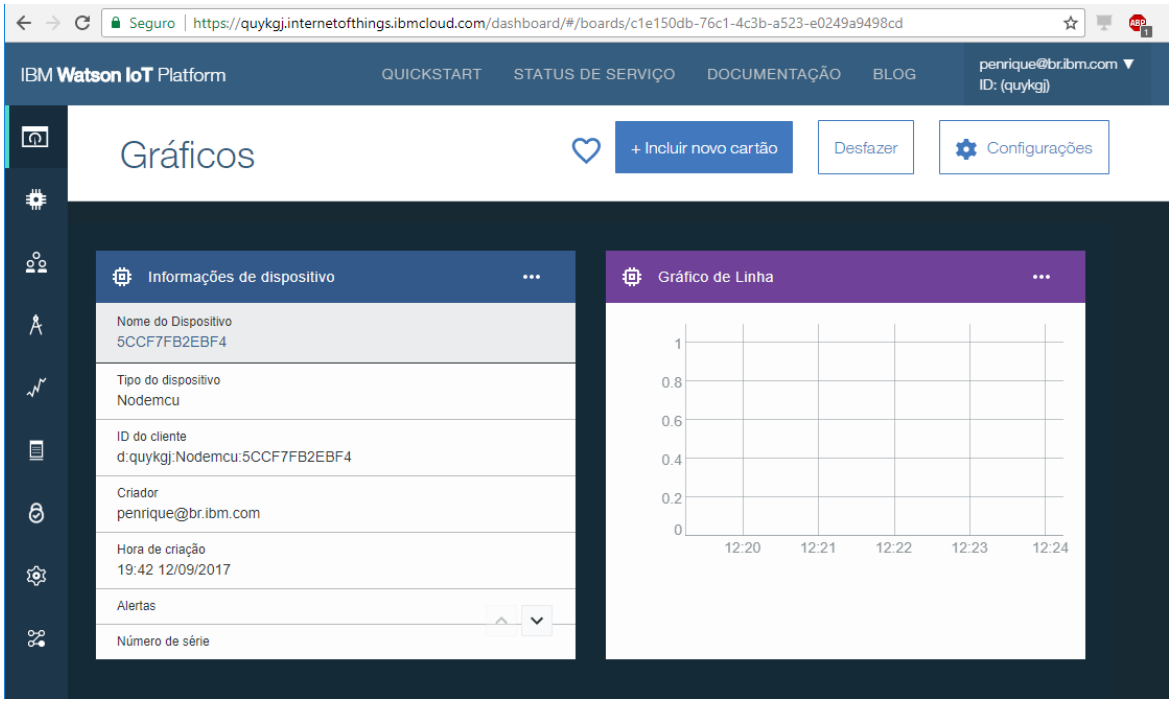

Figura 21 – Conectividade estabelecida.

Fonte: Autoria própria.

Com a conexão estabelecida, a plataforma já está apta a receber dados enviados pelo dispositivo. Como descrito anteriormente, o formato de arquivo enviado pelo Nodemcu é o JSON. A estrutura utilizada pelo JSON para representar informações é simples: para cada valor representado, atribui-se um nome que descreve o seu significado.

No apêndice C, é apresentado o programa carregado no Nodemcu. Ao receber um dado via serial, ocorre uma interrupção. Na rotina de interrupção, é realizada a leitura do dado, transferência para respectivas variáveis e inserção no JSON. Logo em seguida o arquivo é enviado para o broker via MQTT, e uma cópia é enviada para o console de terminal local, para acompanhar o sucesso ou a falha do envio.

Um exemplo do formato JSON enviado para cloud está apresentado a seguir.

```
{
  "d": {
    "Name": "18FE34D81E46",
    "vec_data1": 120,
    "R1": 0,
    "R2": 0,
    "R2n": 0,
    "status": 0,
    "interacao": 10
  }
}
```

3.3.3 Configuração da aplicação no Bluemix

Por meio da tela de configuração de gráficos apresentada anteriormente, é possível criar uma *dashboard* simples para visualização de sensores em geral. Entretanto, o acesso é limitado aos usuários cadastrados na aplicação e não permite personalização. Desta forma, se faz necessário criar uma outra aplicação que permita uma maior flexibilidade e customização. Utilizou-se então o serviço Node-RED.

3.3.4 Node-RED

Node-RED é uma ferramenta cross-plataforma para conectar hardware, APIs, banco de dados e outros serviços online. É baseada na linguagem NodeJS e tem como propósito a rápida prototipação em *Internet of Things*. Dentro da Cloud IBM, é integrada com a plataforma Cloud Foundry, um projeto *open source,* sob licença Apache 2, mantida por meio da contribuição de desenvolvedores e membros da comunidade.

Devido às suas raízes de código aberto, o Cloud Foundry não é específico para o provedor e não o limita a softwares de propriedade intelectual ou infraestrutura de nuvem. O Cloud Foundry extrai a infraestrutura implícita da nuvem para opera-la, permitindo o desenvolvedor se concentrar na elaboração de aplicativos.

Ao criar uma aplicação node-RED pelo Bluemix, automaticamente já é criado um serviço chamado Cloudant NoSQL DB. Trata-se de um banco de dados construído para gerenciar NoSQL JSON. Na aplicação deste trabalho, o Cloudant terá a função de armazenar as configurações do Node-RED, bem como todos os dados coletados pela aplicação de IoT.

A figura 22 resume a arquitetura de integração entre os serviços na cloud IBM.

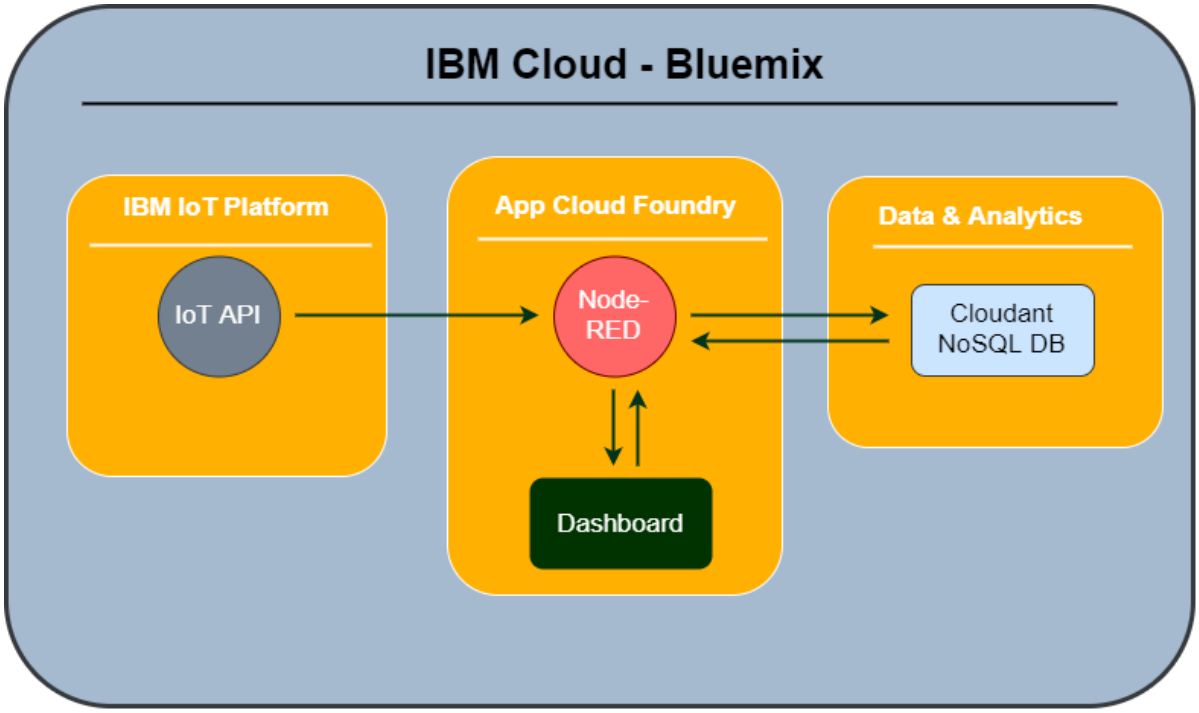

Figura 22 – Arquitetura da aplicação na IBM Cloud.

Fonte: Autoria própria.

Para criar o serviço Node-RED basta acessar o ícone de App Cloud Foundry no catálogo do Bluemix e selecionar o campo de Node-RED Starter. Em seguida, é necessário dar o nome do app e do host, como mostra a figura a seguir.

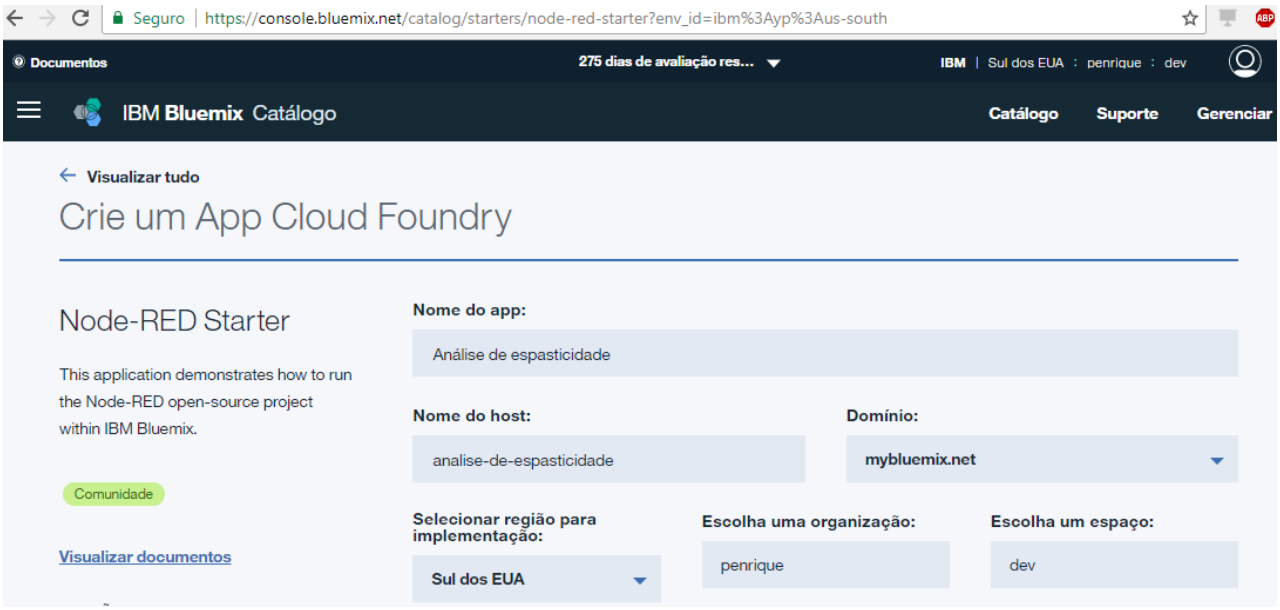

Figura 23 – Node-Red Starter – Configuração inicial.

Fonte: Autoria própria.

Em seguida, pode-se configurar a alocação do serviço, podendo criar outras instâncias e configurar a quantidade de memória desejada.

Figura 24 – Configuração da instância.

Fonte: Autoria própria.

A próxima etapa foi conectar o serviço de IoT Platform, já criado e descrito no item 3.3.2, à aplicação do Node-RED. A conexão entre os serviços é necessária para que haja a comunicação no app Cloud Foundry. A figura 25 mostra a conexão entre os serviços de IoT, Cloudant e de Monitoramento.

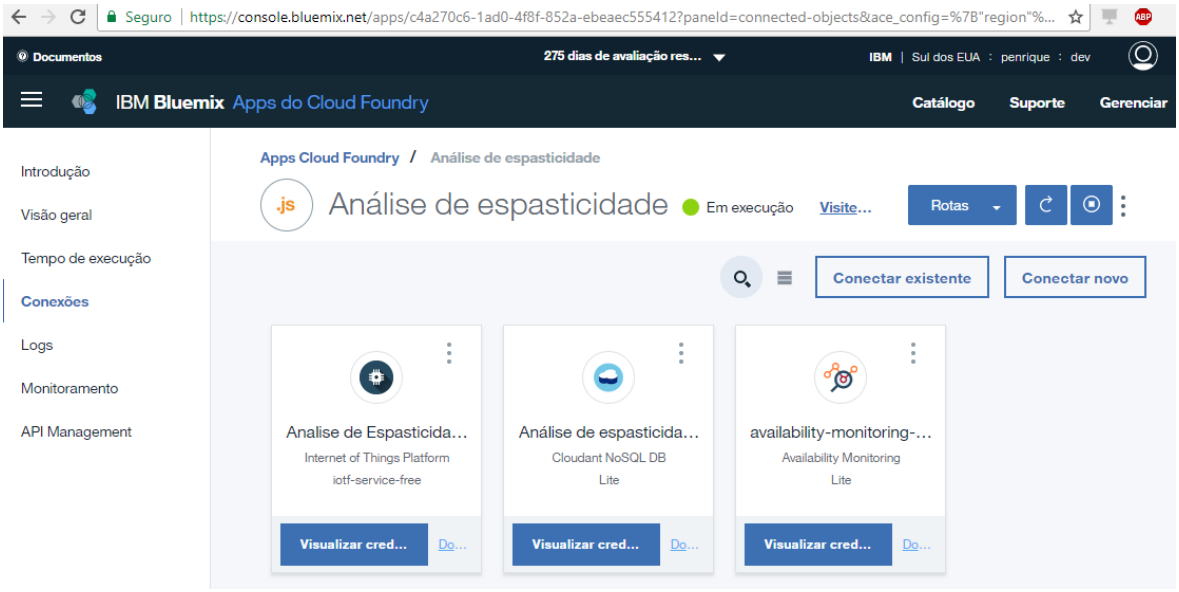

Figura 25 – Conexão entre serviços.

Fonte: Autoria própria.

Com a conexão estabelecida, todo o ambiente de estrutura da aplicação já está configurado. O próximo passo é acessar a URL da aplicação Node-RED para começar o desenvolvimento.

Acessando a URL, visualiza-se o ambiente de desenvolvimento do Node-RED. Os nós encontram-se na lateral esquerda, e para configurá-los basta arrastá-los para área de trabalho.

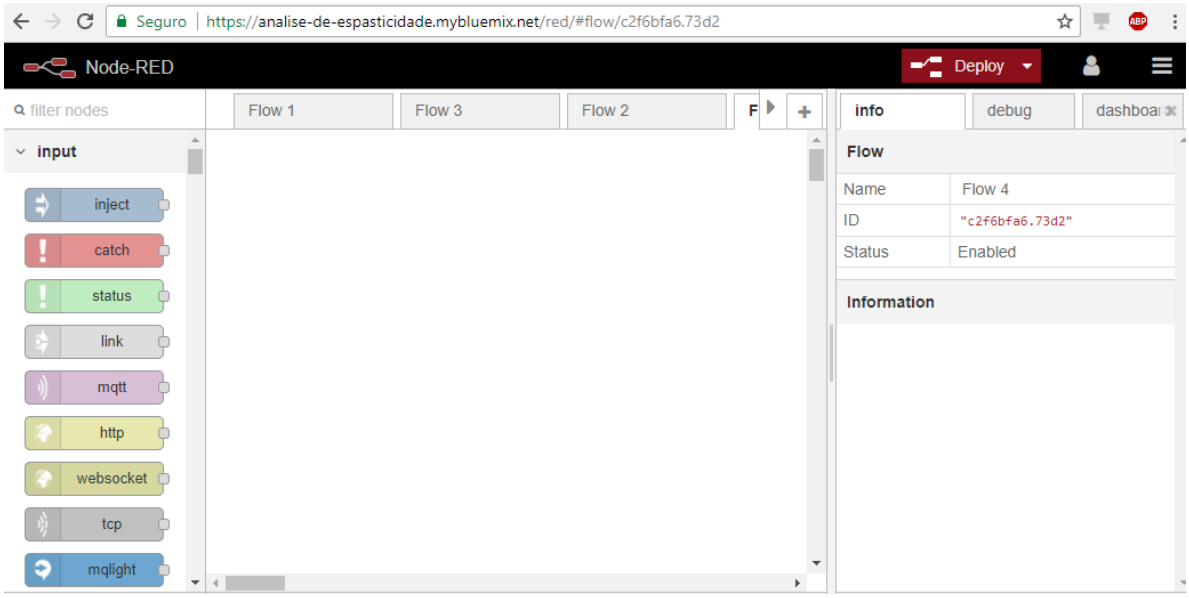

Figura 26 – Ambiente de desenvolvimento Node-RED.

Fonte: Autoria própria.

A figura 27 ilustra o procedimento para configurar o nó de IoT usado nessa aplicação. Como a conexão do serviço já foi estabelecida na configuração da instância Cloud Foundry, o método de autenticação é a opção Bluemix Service. Os demais campos podem ser preenchidos com as informações dos *devices*, como o ID, tipo de evento e formato do arquivo.

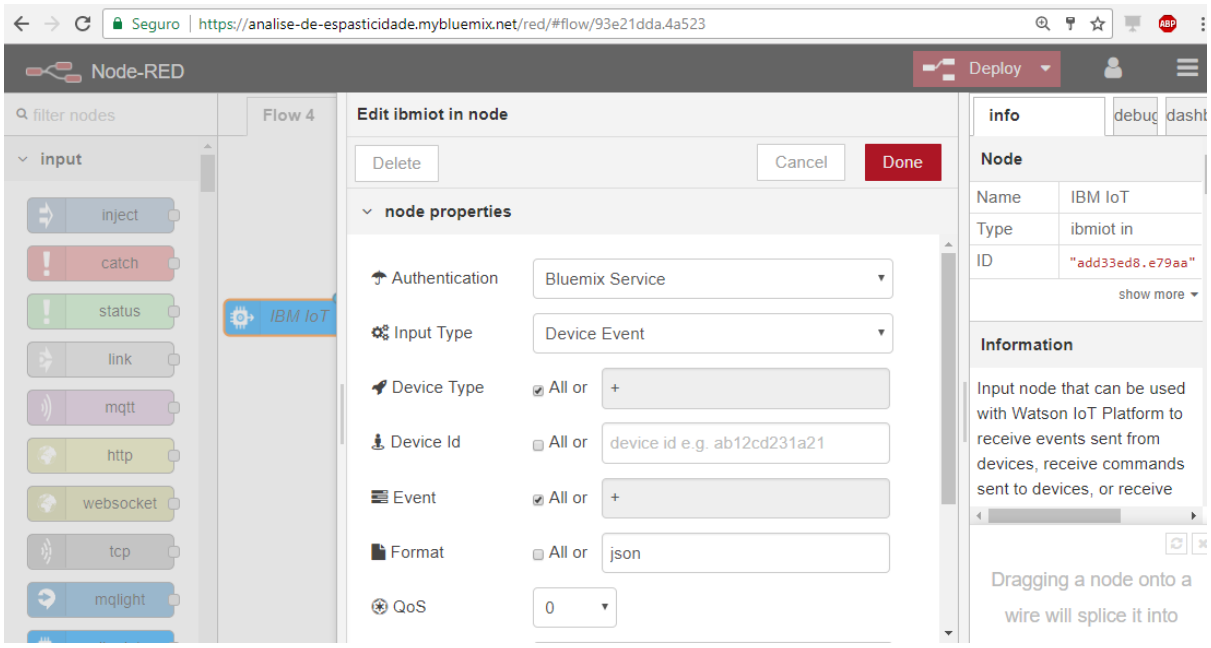

Figura 27 – Configuração do serviço de IoT.

Fonte: Autoria própria.

Outro nó muito utilizado no desenvolvimento deste trabalho foi o de *function.* Com esse nó é possível escrever códigos em *JavaScript* que manipulam um arquivo de entrada e retorna um dado de interesse. A figura 28 apresenta um código simples usado para averiguar se um objeto do JSON de entrada é falso ou verdadeiro, retornando um *msg.payload* diferente para cada situação.

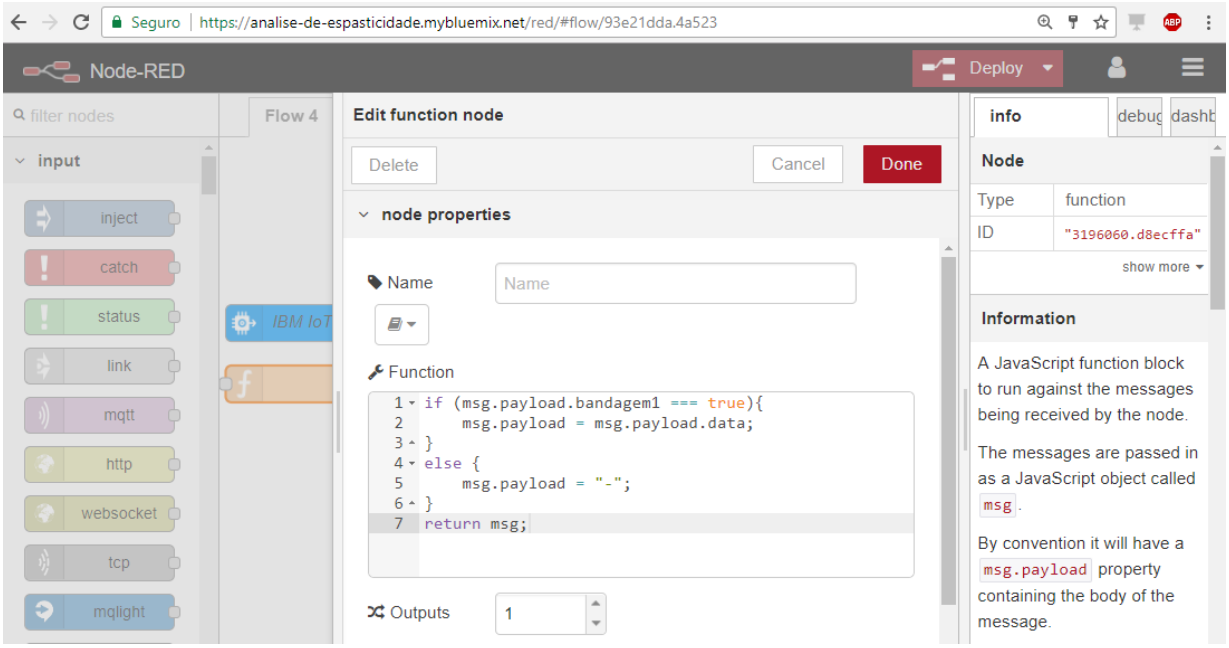

Figura 28 – Configuração de uma função em *JavaScript*.

Fonte : Autoria própria.

Ao acrescentar nós e conexões, é necessário *debuggar* a conexão para concluir se o fluxo está retornando o que era esperado. Para isso, usa-se a função de *debug*, que após realizado o *deploy* da aplicação, permite analisá-la no console na lateral direita do ambiente de desenvolvimento. A figura 29 apresenta a janela de debug de um fluxo responsável pela coleta de texto e inserção no banco de dados.

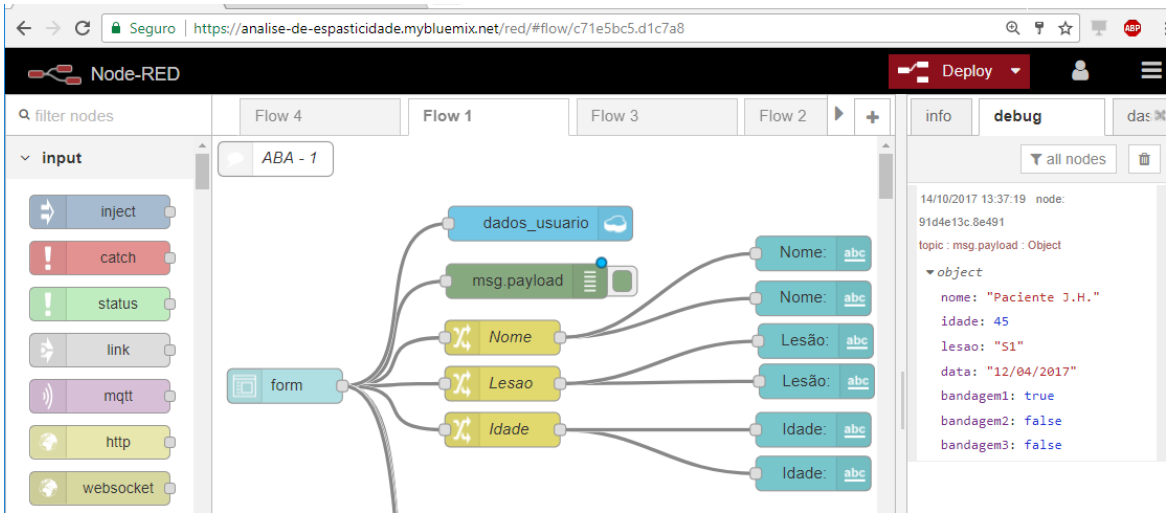

Figura 29 – Exemplo de configuração dos nós e uso do *Debug.*

Fonte: Autoria própria.

Um outro exemplo de fluxo está presente na figura 30. A conexão é responsável pela plotagem em tempo real dos dados recebidos pela API de IoT e pela gravação no banco de dados.

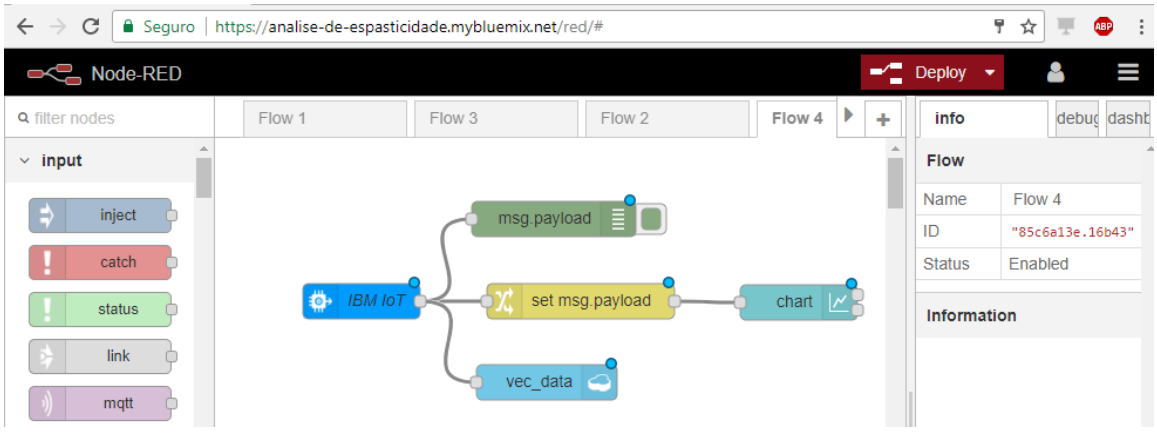

Figura 30 – Conexão entre nó IoT, banco de dados e gráfico.

Fonte : Autoria própria.

Dessa maneira, com as conexões e elaboração dos fluxos, foi possível construir um painel de visualização de maneira simples, versátil e muito funcional. O código do fluxo da

aplicação deste trabalho pode ser encontrado no *github* pelo link. https://github.com/pedrohenriqp/nodered_espasticidade.

Os resultados das configurações e integrações entre hardware e aplicação web estão apresentadas no próximo capítulo.

3.3.5 Coleta dos dados por meio do teste pendular

Os dados de pacientes apresentados neste trabalho foram coletados no Laboratório de Biomecânica e Reabilitação do Aparelho Locomotor da Universidade de Campinas durante a pesquisa intitulada *Contribuição do método Kinesio Taping para a espasticidade em indivíduos com lesão medular*. Os pacientes que participaram do estudo foram convidados a comparecerem à terapia de estimulação elétrica neuromuscular (EENM) durante um período de seis meses para depois participarem do processo da coleta de dados.

Obedecendo protocolos clínicos, os pacientes autorizaram a realização do procedimento por intermédio do termo de consentimento livre e esclarecido.

Primeiramente os indivíduos foram testados quanto a possibilidade de reação alérgica. Não havendo irritação, o tratamento é realizado ao longo de 8 semanas. Durante o processo, receberam semanalmente 10 minutos de EENM em quadríceps femural bilateral, 15 minutos em tibial anterior bilateral e posteriormente aplicações de bandagens do método Kinesio Taping.

Durante o teste pendular, os pacientes estavam sentados em cadeira alta o suficiente para realização do balanço do membro, com encosto reclinado a 60° para promover alongamento do músculo quadríceps, sem provocar grandes alterações de pressão sanguínea (KOS, 2016).

Figura 31 – Teste pendular após aplicação do Método Kinesio Taping.
Fonte: KOS (2016).

Após 8 semanas de acompanhamento com o Método KT, os pacientes foram avaliados novamente para efeito de comparação com os dados coletados no início da intervenção do método KT.

4 Resultados e Discussões

Nesta seção, serão apresentados os resultados obtidos neste projeto seguido de uma discussão.

4.1 Resultados

A utilização de um sistema operacional em tempo real na plataforma embarcada STM32F401 obteve um desempenho adequado O uso de tarefas deixa o código mais limpo e organizado, garantindo uma integridade e determinismo no tempo de execução. O processo de conversão analógico digital também obteve um resultado preciso, como esperado, visto que o processo de leitura e armazenamento no buffer é relativamente simples. O cálculo dos indicadores também se comportou da maneira esperada. A escolha de acrescentar uma tolerância ao algoritmo descrito no capítulo 3 revelou-se uma opção coerente, pois com testes realizados com potenciômetro, uma possível flutuação na leitura poderia causar erros nos cálculos dos parâmetros $A_{0,}$ A_1 e $A_{2..}$A figura 32 apresenta os dados de conversão.

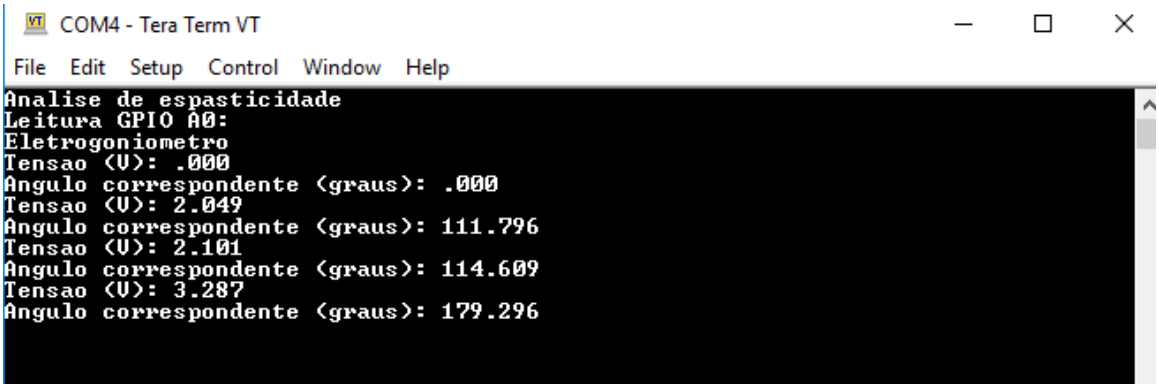

Figura 32 – Leitura do ADC - Terminal serial – STM32F401.

Fonte: Autoria própria.

O dado enviado pelo STM32F401 é então recebido pelo Nodemcu, gerando uma interrupção de serial. Na rotina de interrupção, o dado é inserido em um *payload* em formato JSON que foi transmitido sem problemas para a nuvem via MQTT. Uma vez pré-configurado as credencias do dispositivo, a conectividade entre nuvem e dispositivo não apresentou problema. Em um teste realizado, um vetor de 4000 posições foi enviado sequencialmente sem obter erro no método de *publish.* Também é conveniente lembrar que os procedimentos aqui realizados foram feitos sob um ambiente de internet estável.

A figura 33 apresenta um exemplo da mensagem enviado para a nuvem. Um controle via interface serial conectada ao notebook também se mostrou conveniente para acompanhamento do sucesso ou falha no envio.

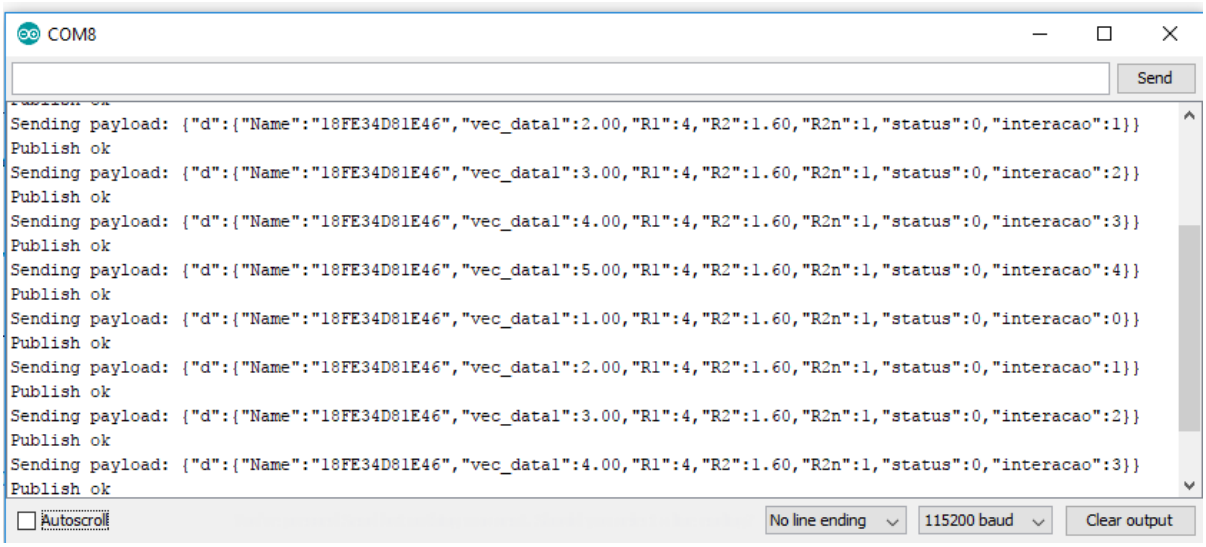

Figura 33 – Envio de payload – Terminal serial - Nodemcu

Fonte: Autoria própria

A plataforma IoT também se comportou da maneira esperada. A identificação e registro de um evento pode ser vista na figura 34, e a descrição dos metadados contido na mensagem pode ser visto na figura 35.

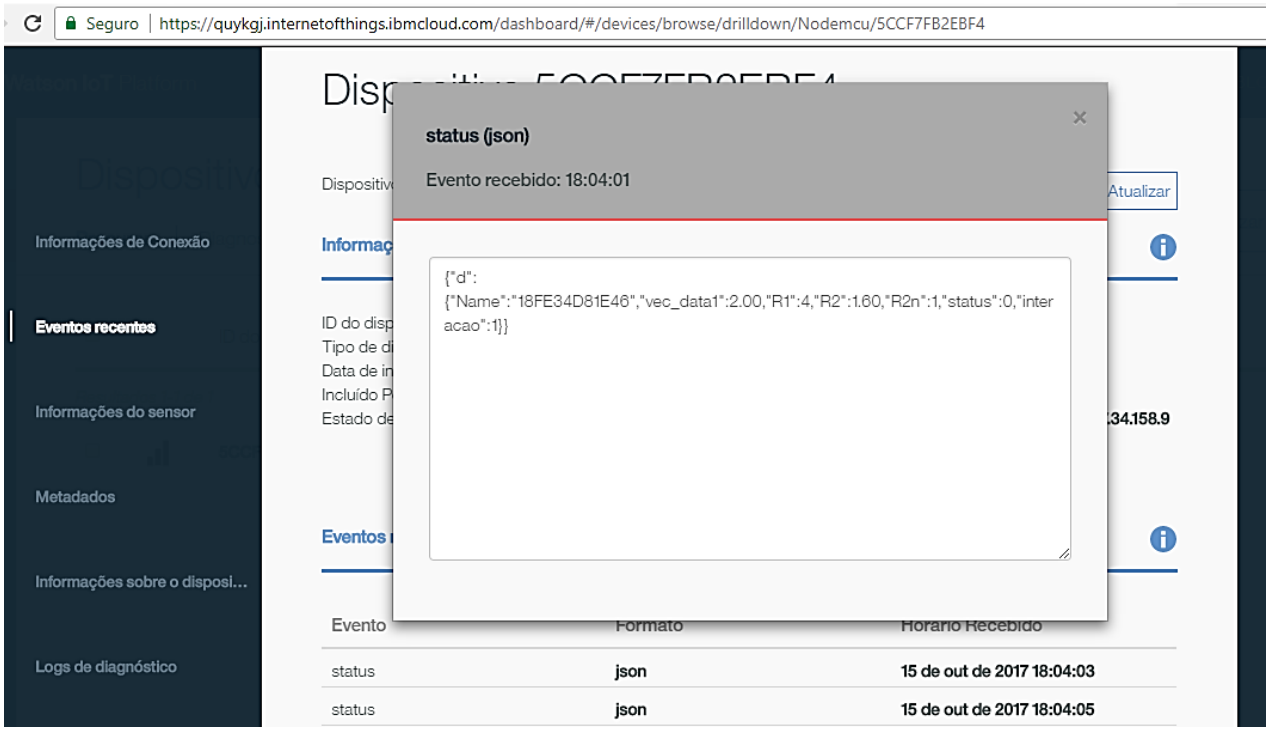

Figura 34 – Recebimento do payload – IBM IoT Plataform.

Fonte: Autoria própria

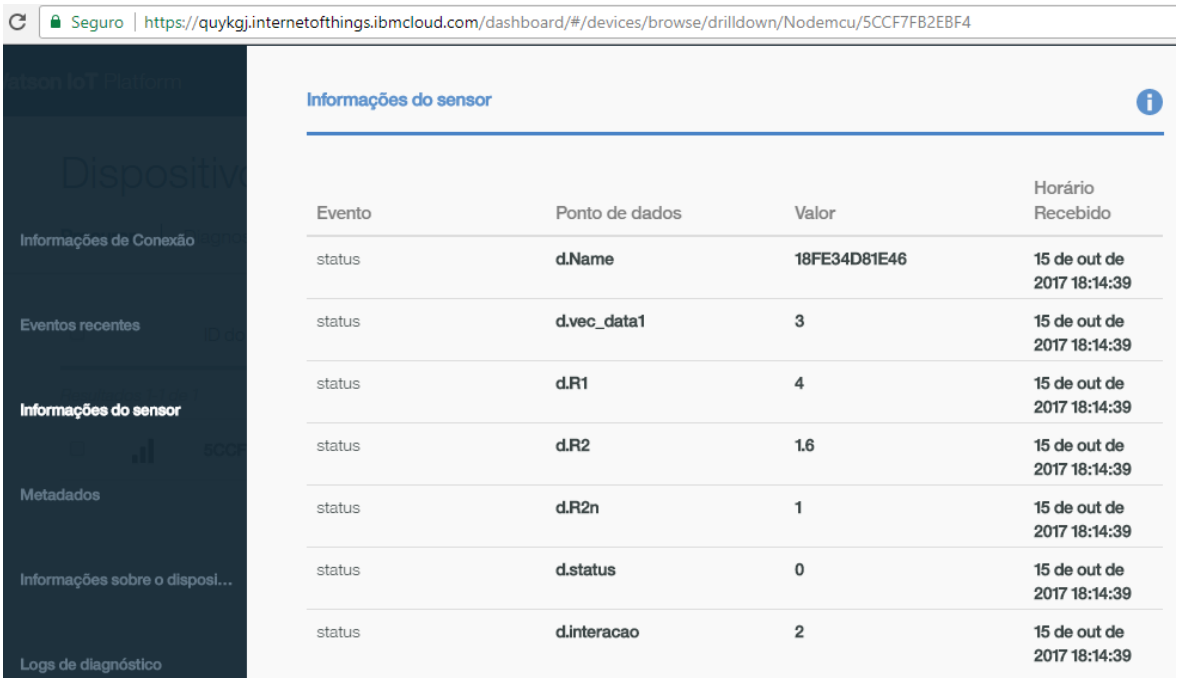

Figura 35 – Descrição de metadados contidos no payload recebido – IBM IoT Plataform.
Fonte: Autoria própria

A aplicação web para exibição dos resultados apresentou grande versatilidade, apesar de ser simples. Possui quatro abas que serão apresentadas a seguir.

A primeira aba, chamada de Dados, coleta informações como o nome do paciente, idade, característica da lesão, data da realização do teste e o número da avaliação a ser realizada. Após o clique em *submit*, os dados são gravados no banco de dados e apresentados nas próximas abas. A figura 36 apresenta a tela inicial da aplicação.

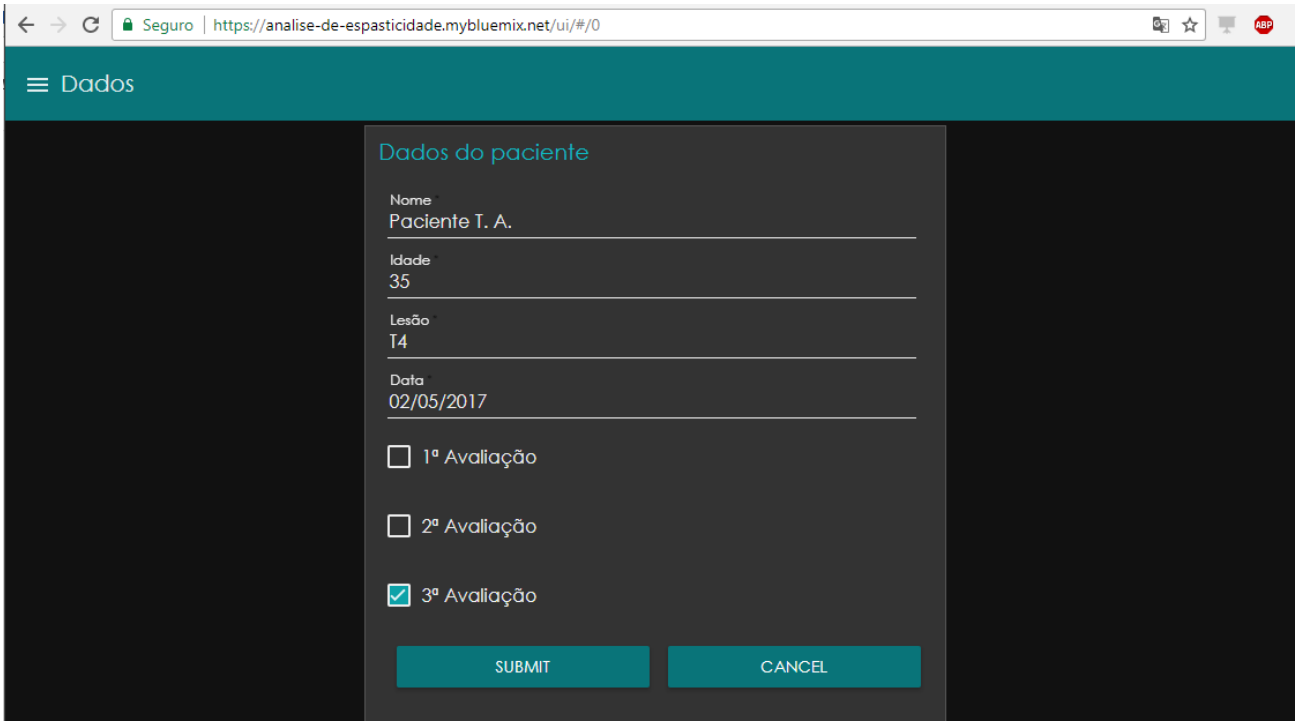

Figura 36 – Tela inicial – Dados.

Fonte: Autoria própria

Ao clicar na lateral esquerda, abre-se o menu para direcionamento de tela, como mostra a figura 37.

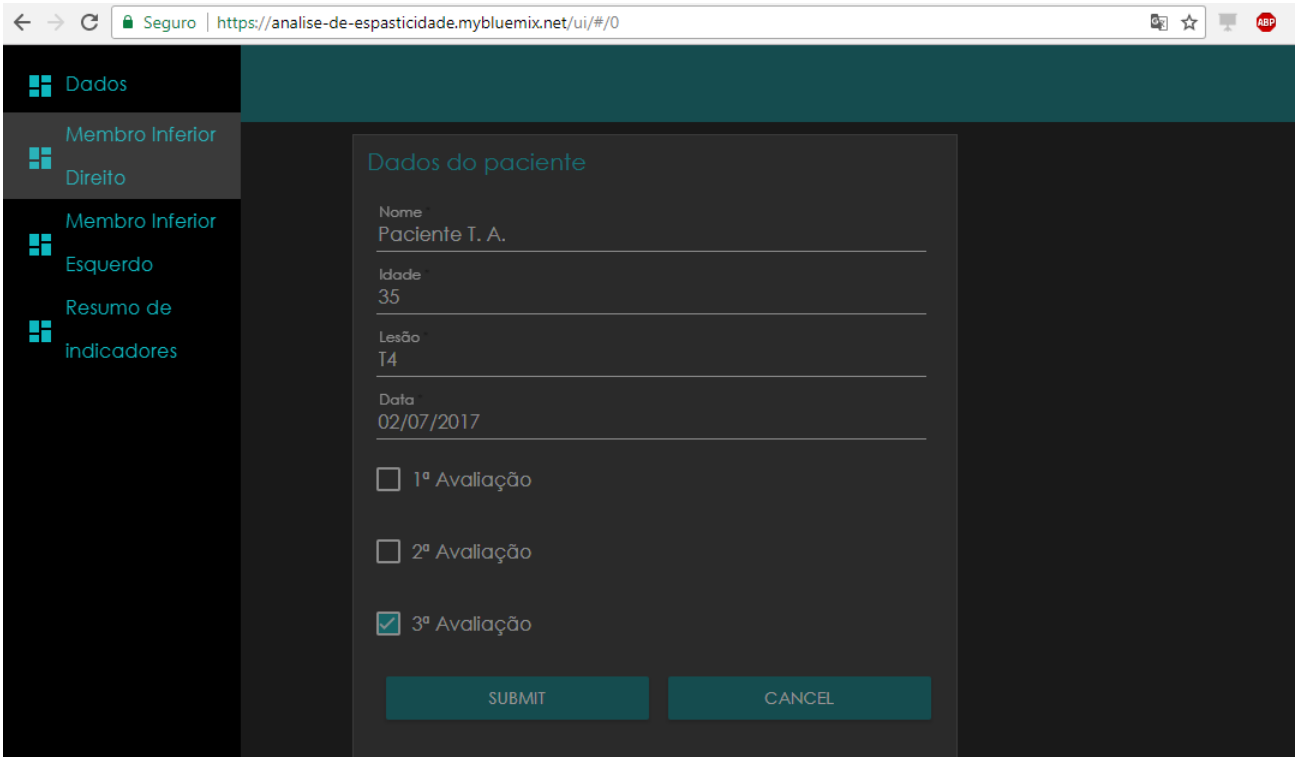

Figura 37 – Menu para direcionamento de tela.

Fonte: Autoria própria

Os dados de um determinado pacientes foram embarcados manualmente em um vetor *float* no dispositivo Nodemcu para serem enviados a nuvem simulando uma captura e envio de uma situação real.

A segunda tela se refere aos dados coletados para o membro inferior direito. Contém dados do paciente, datas das avaliações, controle para início da coleta, *reset* e por fim os gráficos sendo plotados em tempo real. Os indicadores são exibidos após o término da plotagem.

As figuras 38 e 39 exibem a tela para os dados reais de espasticidade, apresentando gráficos com agradável visualização, onde o eixo vertical é dado em graus e o horizontal em tempo, sendo este último correspondente a chegada do dado na API. O intervalo entre a coleta e o tempo de amostragem é configurado na plataforma STM32F401. É conveniente expor que o tempo entre uma coleta e outra não interfere diretamente no cálculo dos parametros R1, R2 e R_{2n}.

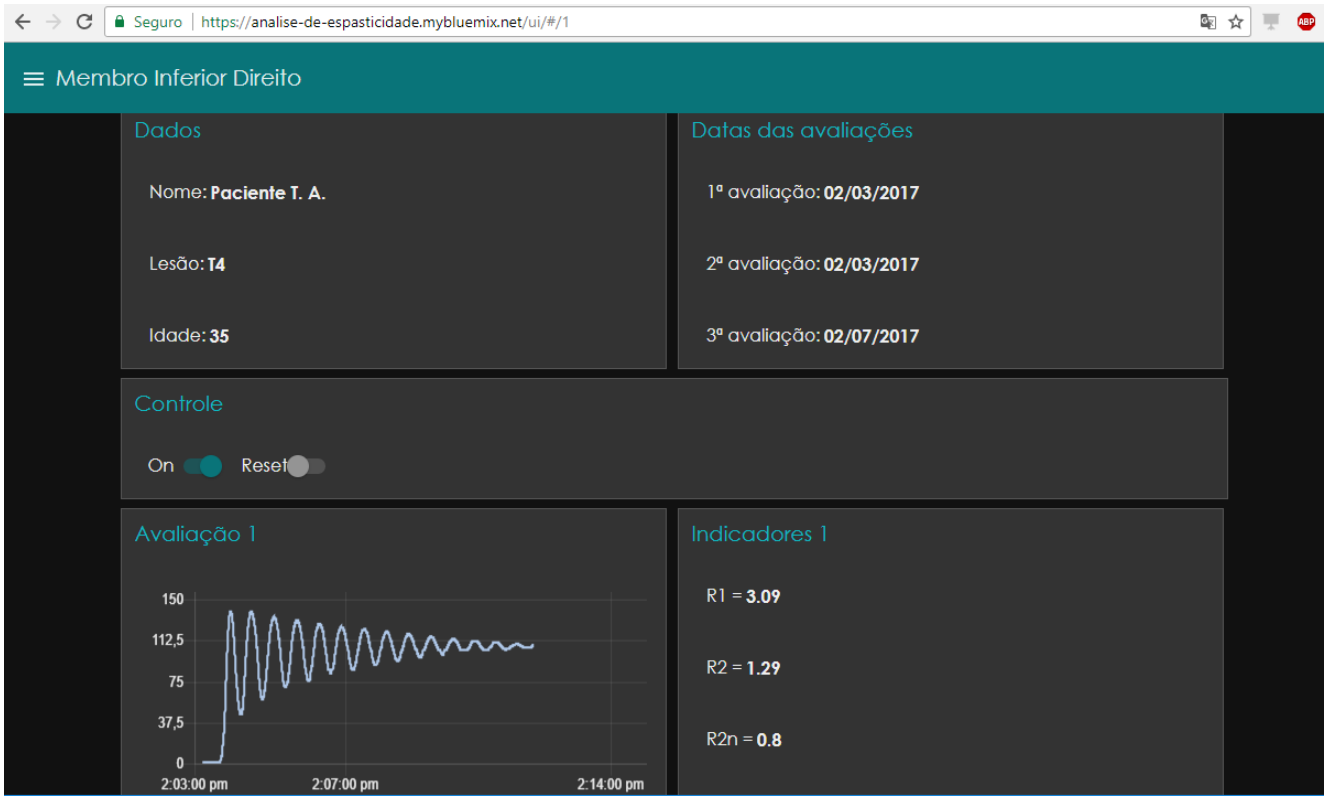

Figura 38 – Dashboard para membro inferior direito.

Fonte: Autoria própria.

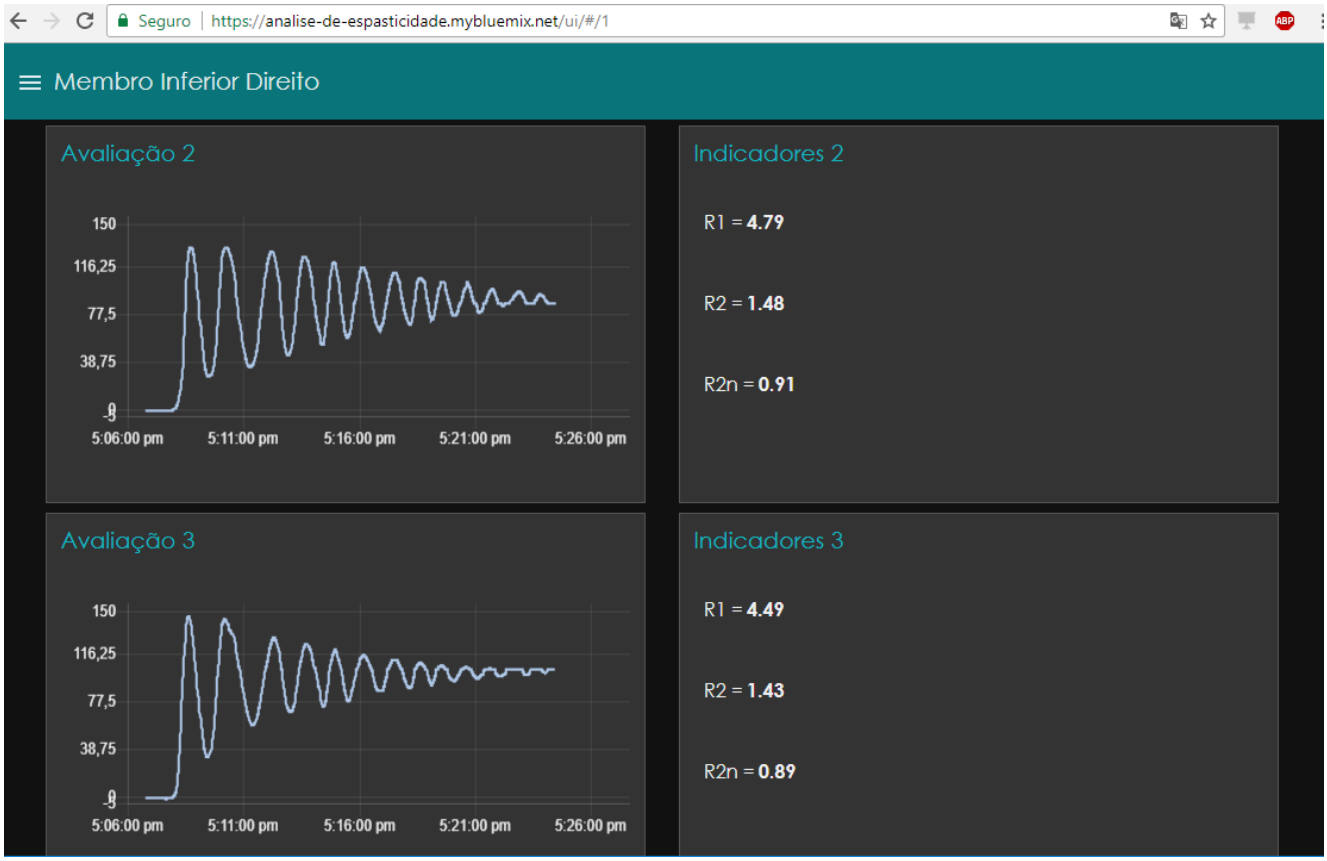

Figura 39 – Dashboard para membro inferior direito – continuação.

Fonte: Autoria própria

A terceira tela se refere ao membro inferior esquerdo, e é composta pelas mesmas informações contidas na aba anterior. Nas figuras 40 e 41 também são apresentados dados reais de paciente onde é possível ter uma aprazível visualização dos resultados.

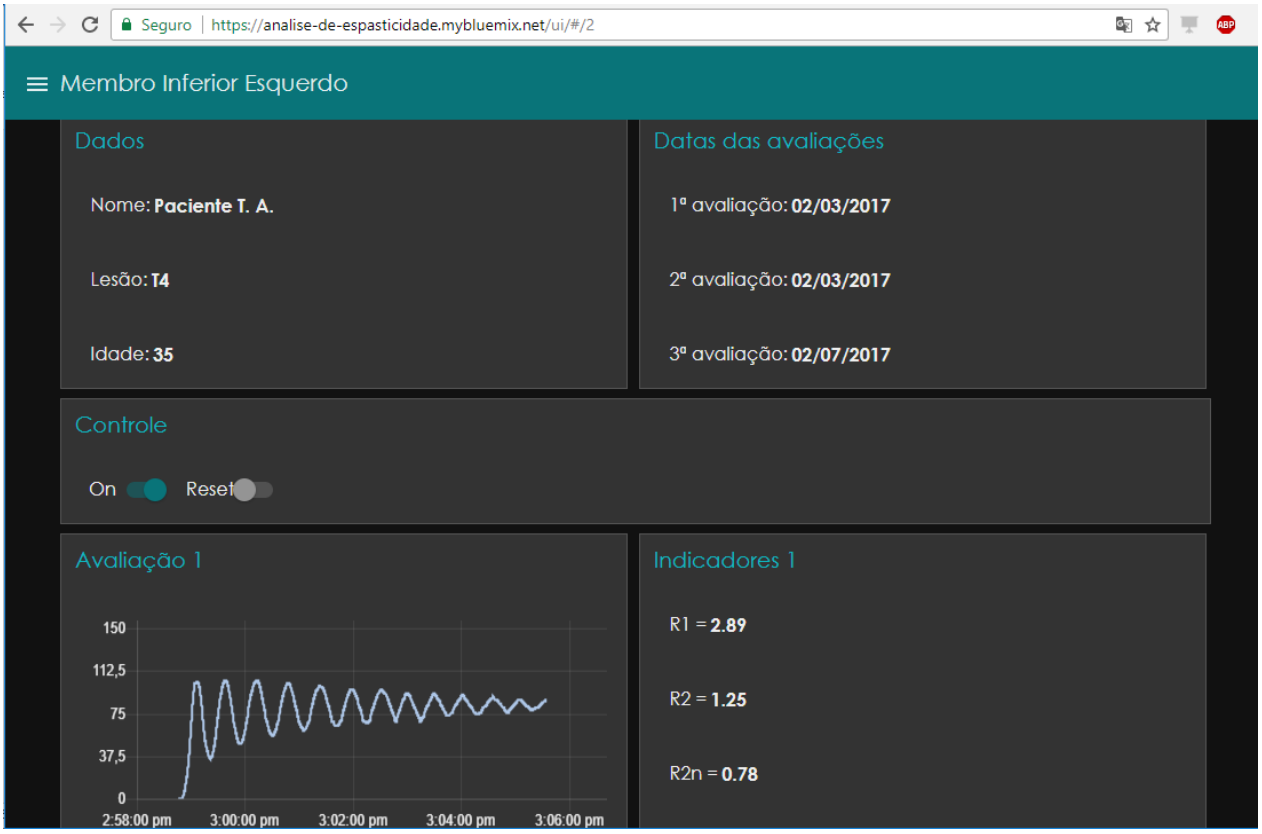

Figura 40 – Dashboard para membro inferior esquerdo.

Fonte: Autoria própria.

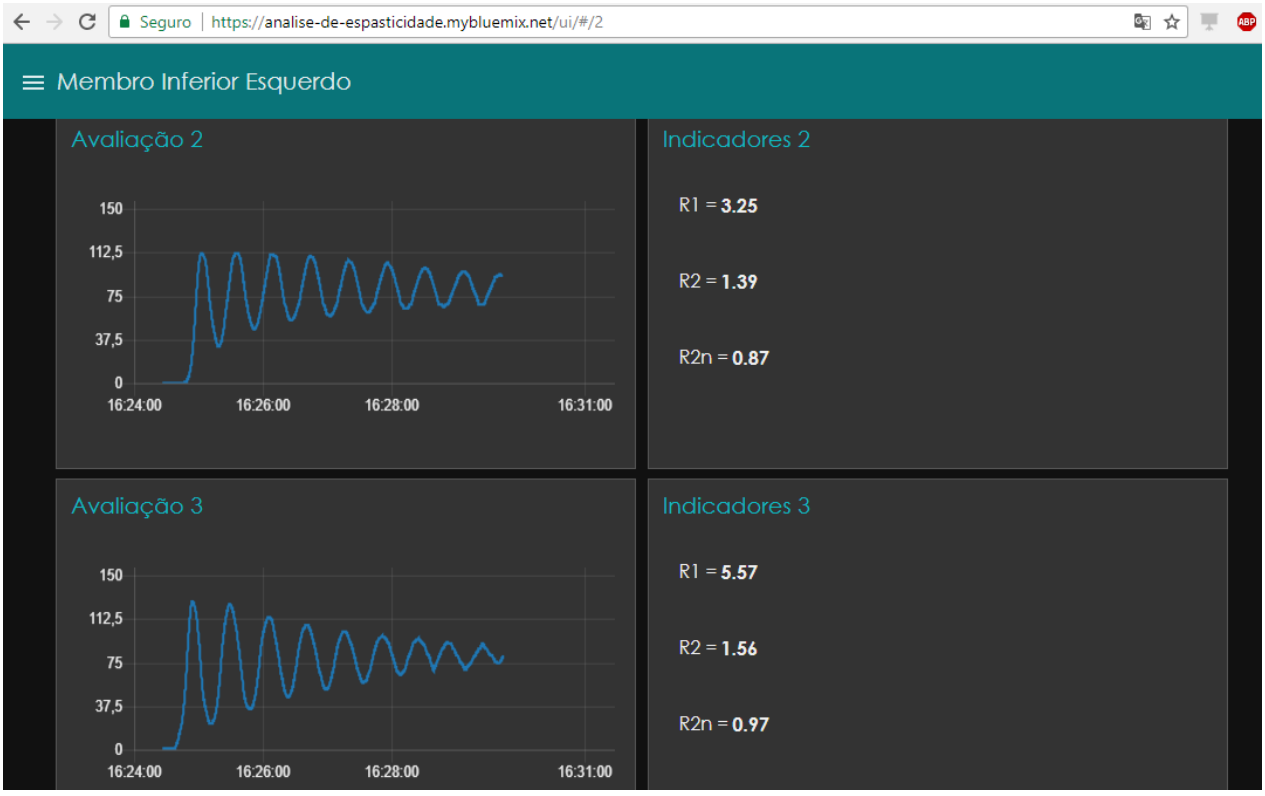

Figura 41 – Dashboard para membro inferior esquerdo – continuação.

Fonte: Autoria própria

Por fim, a quarta e última tela, se refere a um resumo dos indicadores realizados nas 3 avaliações, tanto do membro inferior direito, quanto esquerdo. É apresentado um gráfico ilustrativo do teste pendular e como os parâmetros foram calculados. Também são indicadas as datas das avaliações.

A aba apresenta duas tabelas que resumem os indicadores apresentados nas abas anteriores e sinaliza o status de avaliação de acordo com a mudança dos indicadores nas avaliações. O status verde sinaliza que o indicador apresentou uma melhora desde a primeira avaliação. O status laranja sinaliza que o parâmetro apresentou instabilidade quanto a evolução e o status vermelho indica que ocorreu uma piora. A indicação de status é feita de forma automática e serve como orientação para uma conclusão clínica da evolução de um tratamento, neste caso o método KT.

A terceira tabela resume a representatividade e os valores de referência para cada parâmetro, seguido de um *link* para o artigo de Badj e Vodovnik para uma consulta mais completa do significado dos resultados obtidos. A figuras 42 e 43 apresentam a tela de resumo de indicadores.

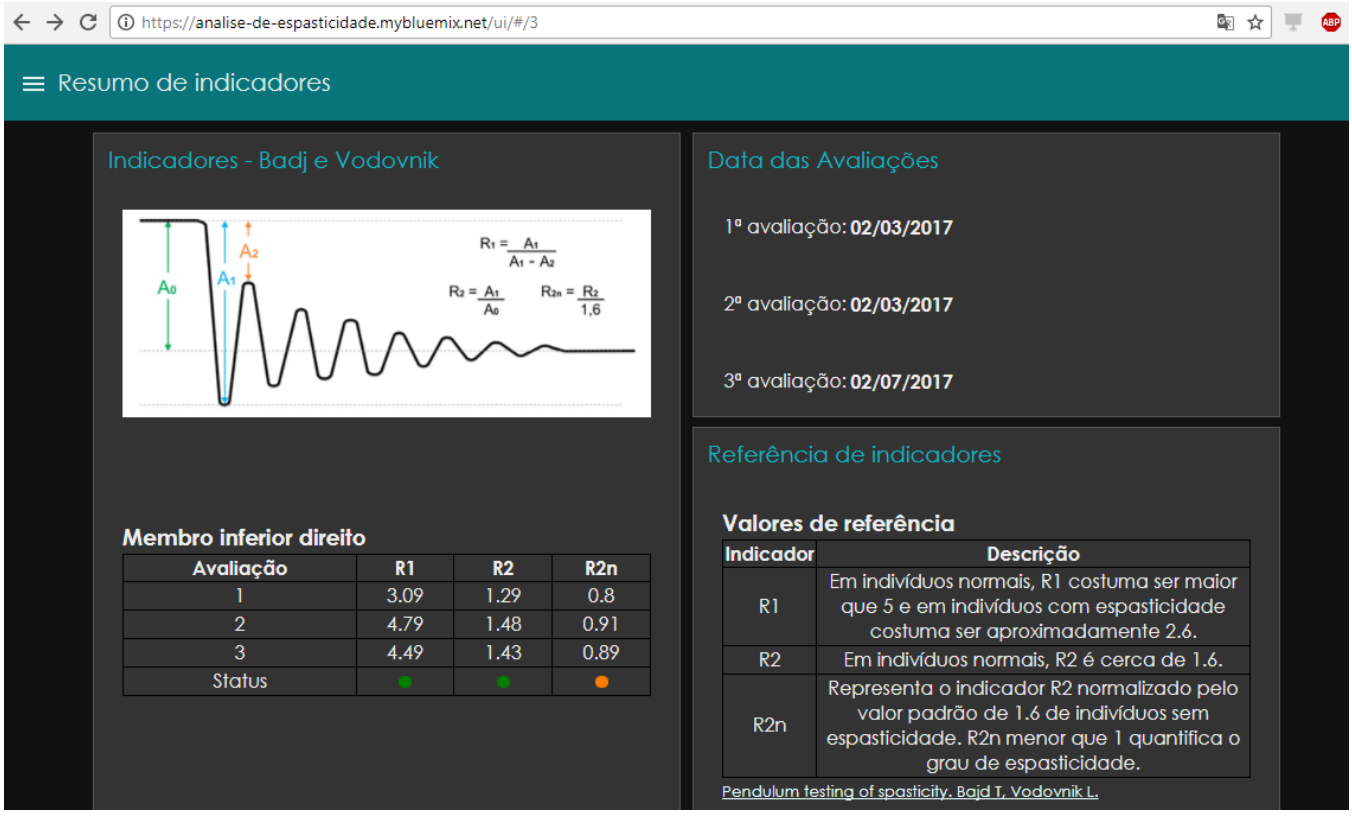

Figura 42 – Resumo de indicadores.

Fonte: Autoria própria.

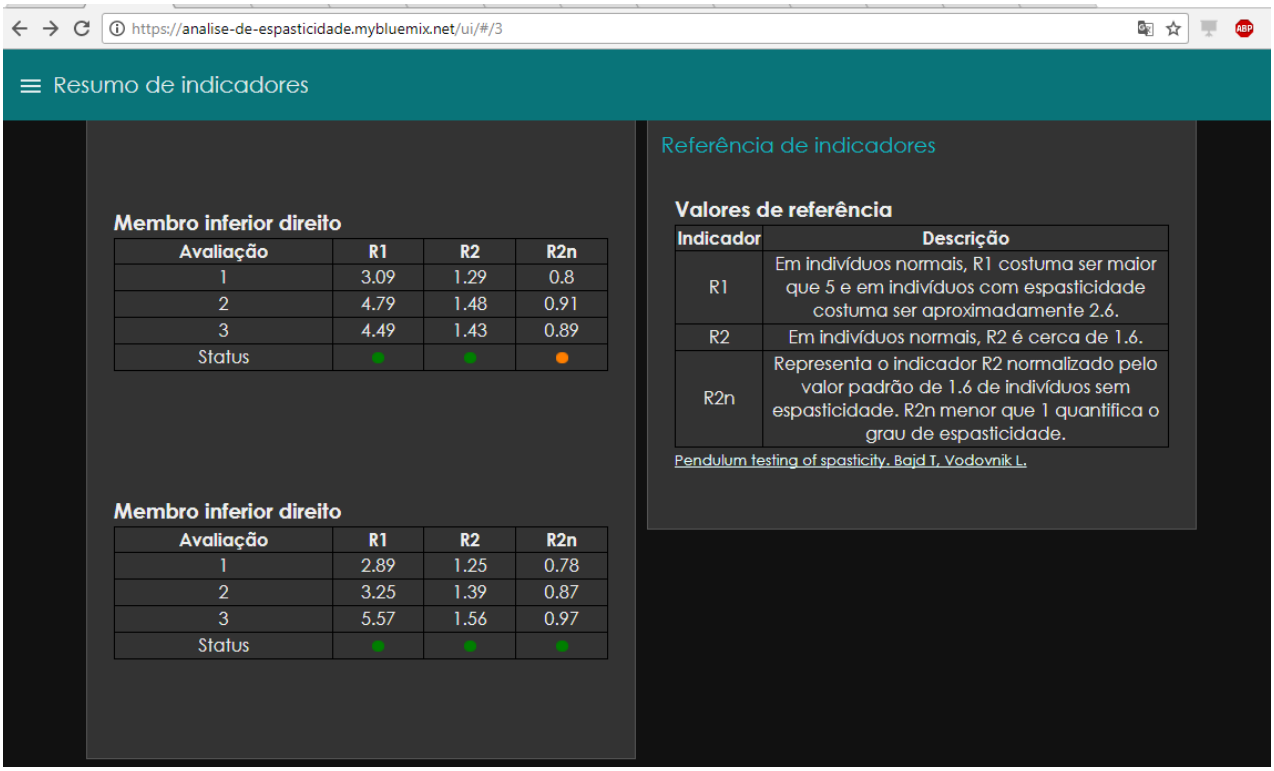

Figura 43 – Resumo de indicadores – continuação.

Fonte: Autoria própria

Por meio dos dados reais apresentados no conjunto de figuras anteriores, percebe-se uma melhora nos parâmetros, indicando uma possível diminuição da espasticidade do paciente por meio do tratamento *Kinesio Taping*. Ressalta-se a importância da validação médica sobre os resultados apresentados pelo sistema, servindo ao ambiente de monitoramento apenas captar e apresentar os gráficos obtidos, bem como fornecer a evolução dos indicadores.

4.2 Discussão

A Plataforma STM32F401, portando o freeRTOS, apresentou um comportamento bastante significativo, apesar da simplicidade relativa das tarefas. O algoritmo para detecção de picos, vales e regime estacionário não manifestou cálculos complexos, motivo pelo qual não ser necessário a implementação de instruções de DSP. Como o sinal de estudo possui uma forma conhecida, semelhante a uma senóide amortecida, o uso de uma margem de tolerância para localizar os pontos de interesse mostrou-se fundamental. A maioria das flutuações no sinal proveniente de oscilações e/ou ruído são ignoradas no levantamento dos parâmetros. No entanto, alguns testes com muitas flutuações consecutivas apresentaram uma instabilidade na identificação do ponto de máximo e mínimo. Uma alternativa a essa manipulação de dados é transferi-los em estados originais para a nuvem, onde então seria

conveniente suavizar o conjunto de dados para detectar os pontos de máximo e mínimo de interesse.

O dispositivo Nodemcu também obteve um comportamento condizente com o esperado. Um procedimento realizado para teste conseguiu enviar initerruptamente 5000 valores para a plataforma IBM IoT com sucesso. O tempo de envio e recebimento do json via MQTT, entretanto, apresentou um atraso. Indícios levam a duas explicações possíveis: *delay* no envio do *payload* devido a uma compilação não otimizada da biblioteca de *publish/subscribe* e/ou *delay* na resposta de sucesso/falha do *broker*. Tais motivos estimulam o estudo de outros métodos para conectividade de dados a nuvem.

A aplicação web em forma de *dashboard* interativa também obteve um resultado adequado. A plataforma IBM de IoT mostrou-se ser uma excelente ferramenta para captar dados de sensores e disponibilizá-las para demais aplicações. O uso do Node-RED otimizou o tempo de elaboração de um *dashboard*, fornecendo condições de construir uma aplicação simples, porém com grande eficácia na comunicação entre APIs. A conexão com o banco de dados Cloudant não apresentou problemas, entretanto com o aumento de pacientes e números de teste, observou-se que é conveniente construir uma busca a base de dados de maneira mais responsiva, podendo existir uma seção para cada paciente, com login e senha.

Convém lembrar que a integração do sistema foi validada com o uso de um potenciômetro simulando o eletrogoniômetro. Em uma aplicação real, seria conveniente calibrar o processo de conversão de tensão-ângulo especificamente para cada paciente, ajustar o tempo total de coleta de amostragem e possivelmente aplicar algum tipo de filtro digital dependendo da existência de ruído no sinal proveniente do eletrogoniômetro.

Com a discussão e os resultados expostos, o ambiente para análise de espasticidade construído neste trabalho apresentou ser uma excelente ferramenta de auxílio clinico, integrando sinais físico, hardware e computação em nuvem.

5 Conclusão

Este projeto teve como objetivo o desenvolvimento de um ambiente para avaliação de espasticidade utilizando um sistema embarcado e computação em nuvem. O trabalho contemplou todas as etapas de um projeto, contando com uma abordagem teórica, identificação da solução, desenvolvimento, testes e reconhecimento das limitações do sistema.

O teste pendular, apesar de simples e antigo, ainda se configura em uma maneira conveniente na avaliação da desordem motora e no aumento do tônus muscular desenvolvidos pela lesão medular. Os parâmetros apontados por Badj e Vodovonik quantificam e qualificam o grau de espasticidade em relação a indivíduos sem espasticidade, servindo como indicadores ao longo de um tratamento do distúrbio.

A utilização da plataforma STM32F401 portando o sistema operacional em tempo real FreeRTOS mostrou-se ser uma excelente opção a microcontroladores convencionais de 8 bits, com melhor custo/benefício, robustez e confiabilidade na aplicação, podendo agregar tarefas mais complexas a serem executadas no sistema embarcado. O dispositivo Nodemcu se comportou da maneira adequada, com a restrição no tempo entre os envios de payloads para a nuvem, como observado na seção de resultados e discussões.

A plataforma IBM Watson IoT representou uma excelente maneira de receber dados de dispositivos eletrônicos, fornecendo chamadas de API para utilização em demais aplicações, como o Node-RED. A *dashboard* construída por meio do *app Cloud Foundry* se configurou em uma maneira simples e útil na criação de painéis de visualizações, otimizando o tempo de integração enrte plataformas e fornecendo uma visualização em tempo real na construção de gráficos.

Com o exposto, pode-se concluir que a abordagem do uso de IoT no contexto clínico é de grande potencial, especialmente em procedimentos que necessitam de um monitoramento de sinais, sejam físicos ou biológicos. Dessa forma, o ambiente para análise de espasticidade em lesados medulares desenvolvido neste trabalho pode ser considerado como uma excelente ferramenta de auxílio em tratamentos fisioterapêuticos, integrando sinais físicos, sistemas embarcados e computação em nuvem.

5.1 Trabalhos futuros

Algumas melhorias podem ser feitas em trabalhos futuros. Para fornecer maiores informações sobre o teste pendular é conveniente agregar ferramentas de *analitycs* para obter *insights* sobre o distúrbio e analisar os demais indicadores do teste pendular presentes nos

artigos de Bajd e Vodovnik. Aumentar a base de dados e utilizar algoritmos para reconhecimento de padrões e aprendizado de máquina para obter possíveis predições. Estudar outro método de comunicação entre sistema embarcado e nuvem, como o protocolo CoAP e as tecnologias Bluetooth e ZigBee. Deixar a plataforma com melhor conectividade com o banco de dados, podendo criar uma área para cada paciente, facilitando o acesso ao histórico e ampliar a *dashboard* para outras disfunções.

REFERÊNCIAS

ARAUJO, R.; KURTHY, C. Fisioterapia e lesão medular. **Associação Brasileira de Fisioterapia Neurofuncional**, 2015. Disponível em: <http://abrafin.org.br/wp-content/uploads/2015/01/LESAO_MEDULAR.pdf>. Acesso em: 02 abr. 2017.

AZEVEDO, E. R. F. B. M. **Avaliação da espasticidade e da composição corporal de indivíduos com lesão medular crônica**. 2015. 93f. Tese (Doutorado em Ciências da Cirurgia) – Faculdade de Ciências Médicas, Universidade Estadual de Campinas, 2015. Disponível em: < http://repositorio.unicamp.br/handle/REPOSIP/312464>. Acesso em: 25 mar. 2017.

BAJD, T.; VODOVNIK, L. Pendulum testing of spasticity. **J Biomed Eng**, vol. 6, n. 1, p. 9-16, jan, 1984. Disponível em: < https://www.ncbi.nlm.nih.gov/pubmed/6694374>. Acesso em: 02 abr. 2017.

BOHANNON, R. W.; SMITH, M. B. Interrater reliability of a modified Ashworth scale of muscle spasticity. **Phys Ther**., v. 67, n. 2, p. 206-7, feb. 1987. Disponível em: < https://www.ncbi.nlm.nih.gov/pubmed/3809245>. Acesso em: 02 abr. 2017.

COSTA, A. J. **Característica de Transmissão: Atenuação e Dispersão**. Disponível em: <https://paginas.fe.up.pt/~hsalgado/co/como_03_atenuacaoedispersao.pdf>. Acesso em: 29 jul. 2017

DANGELO, J. G; FATTINI, C. A. **Anatomia Humana básica.** 2. ed. São Paulo: Editora Atheneu, 2002.

DONNO, M. et al. A New Flexible Optical Fiber Goniometer for Dynamic Angular Measurements: Application to Human Joint Movement Monitoring. **IEEE**, vol. 57, n. 8, aug. 2008. Disponível em: <http://edge.rit.edu/edge/P10010/public/2TestMethods/opticalfibergoniometer.pdf/>. Acesso em: 29 jul. 2017.

FOWLER, E. G.; NWIGEW A. I., HO T. W. Sensitivity of the pendulum test for assenssing spacitic y in person with cerebral palsy., **Dev Med Child Neurol**, v. 42, n. 3, p. 182–189, Mar. 2000. <http://onlinelibrary.wiley.com/doi/10.1111/j.1469-8749.2000.tb00067.x/epdf>. Acesso em: 26 ago. 2017.

IFSC- Instituto Federal de Santa Catarina. **Sistemas Ópticos: Atenuação e Dispersão**. Santa Catarina, 2015. Disponível em: <http://www.sj.ifsc.edu.br/~saul/sistemas%20opticos/SIO_4_atenuacao_%202015-2.pdf>. Acesso em: 29 jul. 2017.

ISLAM, S. M. R.; KWAK, D.; KABIR, M. H.; HOSSAIN, M.; KWAK, K. S. The Internet of Things for Health Care: A Comprehensive Survey. ***IEEE Access***, vol. 3, p. 678-708, 2015. Disponível em: < http://ieeexplore.ieee.org/document/7113786/all-figures?reload=true#PageTop>. Acesso em: 16 set. 2017.

KÓS, R. S. **Contribuições do método Kinesio Taping para a espasticidade em indivíduos com lesão medular**. 2016. 28f. Projeto de Doutorado para obtenção do título de Doutor (Doutorado em Ciências da Cirurgia) - Faculdade de Ciências Médicas, Universidade Estadual de Campinas, 2015.

LEMOS, T; DIAS, E. **Kinesio Taping: Introdução ao Método e Aplicações Musculares**. São Paulo: Andreoli, 2013.

MARIA, R. M. **Aprimoramento e aplicação de um aparelho para avaliação de espasticidade em lesados medulares**. 2015. Dissertação (Mestrado em Ciência da Cirurgia) - Faculdade de Ciências Médicas, Universidade Estadual de Campinas, 2015.

MARIA, R. M. **Desenvolvimento de um sistema para avaliação de espasticidade em lesados medulares**. 2009. 56f. Monografia (Conclusão de Curso de Engenharia com ênfase em Eletrônica) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2009.

BRASIL. MINISTÉRIO DA SAÚDE. **Diretrizes de atenção à pessoa com lesão medular**. 2.ed. Brasília - DF, 2013.

NITRINI, R.; BACHESCHI, L. A. **A neurologia que todo médico deve saber**. 2. ed. São Paulo: Editora Atheneu, 2003.

PEREIRA, R. J. G. **Fibras ópticas e WDM**. Rio de Janeiro, 2008. Disponível em: <https://www.gta.ufrj.br/grad/08_1/wdm1/Atenuaoelimitaesdasfibraspticas.html>. Acesso em: 29 jul. 2017.

PRADO, S. **Sistema de Tempo Real – Parte 1**. 2010. Disponível em: <https://sergioprado.org/sistemas-de-tempo-real-part-1/>. Acesso em: 12 ago. 2017.

REDE SARAH DE HOSPITAIS DE REABILITAÇÃO. **Lesão medular: consequências e tratamentos**. Distrito Federal, 01 jan. 2010. Disponível em: <http://www.bengalalegal.com/medular#s4>. Acesso em: 05 ago. 2017.

RINA, R.; ICARANGAL, A.; BERTELLOTT, G. **Spasticity and Spinal Cord Injury**. University of Washington – Rehabilitation Medicine. Washington, jan 2015. Disponível em: < http://sci.washington.edu/spasticity/>. Acesso em: 05 ago. 2017.

SHIRATSU, A.; COURY, H. J. C. G. Reliability and accuracy of different sensors of a flexible electrogoniometer. **Clinical Biomechanics**. São Carlos. v.18, n. 7, p. 682-684, aug, 2003.

SOCIEDADE PAULISTA DE MEDICINA FÍSICA E REABILITAÇÃO. **Espasticidade: Conceitos atuais baseados em evidências científicas**. 1.ed. São Paulo, 2004. Disponível em: <https://toneurologiaufpr.files.wordpress.com/2013/03/espasticidade-conceitos-atuais.pdf>. Acesso em: 05 ago. 2017.

TAMBURELLA, F; SCIVOLETTO, G; MOLINARI, M. Somatosensory inputs by application of Kinesio Taping: effects on spasticity, balance, and gait in chronic spinal cord injury. **Front Hum Neurosci.**, v. 8, may 2014 2014. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/24910607>. Acesso em: 07 out. 2017.

TEIXEIRA, L. F., OLNEY, S. J., BRAUWER, B. Mecanismos e medidas de espasticidade. **Rev Fisioter**, São Paulo, v. 5, n. 1, p. 4-19, jan./jun., 1998. Disponível em: < http://www.revistas.usp.br/fpusp/article/view/76781/80643>. Acesso em: 16 set. 2017.

TEIXEIRA-SALMELA L. F. et al. Pêndulo: um teste simples de medida de espasticidade. **Rev Acta Fisiátrica**, Minas Gerais, v.9, n. 2, p. 63-70, ago. 2002. Disponível em: < http://www.revistas.usp.br/actafisiatrica/article/view/102364>. Acesso em: 16 set. 2017.

VARGA, S.; ROTTA, T. C. **Afinal, o que é computação cognitiva?**. IBM Academy of Technology Affiliated. 2016. Disponível em: <https://www.ibm.com/developerworks/community/blogs/tlcbr/resource/mp/TLC-BR_Mini_Paper_Ano_11N_270_Computacao_cognitiva.pdf?lang=en>. Acesso em: 07 out. 2017.

WINTER, D. A. Concerning the scientific bais for the diagnosis of pathological gaits and for rehabilition protocols. **Phys Ther**., v. 37, p. 245-52, jan. 1985. Disponível em:

<http://www.jbiomech.com/article/0021-9290(90)90044-4/references> . Acesso em: 16 set. 2017.

YANBING, L. I.; POTKONJAK, M.; WOLF, W. Real-time operating systems for embedded computing. **IEEE**. p. 388-392, oct. 1997. Disponível em: <http://web.cs.ucla.edu/~miodrag/papers/Li_ICCD_97.pdf>. Acesso em: 09 ago. 2017.

APÊNDICE A – CÓDIGO– STM32F401 -RTOS

```
/**********************************************************************
**
  * File Name          : main.c
  * Project            : Eletrogoniometro - Análise de espasticidade
  * Description        : Main program body
  * Author             : Pedro Henrique Pereira
  * Version            : 1
  * Year               : 2017
  *
  *This code was configured by the author using STM32F4 HAL driver and
FreeRTOS through STMCubeMX- STMicroelectronics International N.V.

**********************************************************************
**/

/* Includes ------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_hal.h"
#include "cmsis_os.h"
#include "string.h"
#include "stdlib.h"
#include "float_to_string.h"


/* Private variables ---------------------------------------------------
*/
ADC_HandleTypeDef hadc1;
UART_HandleTypeDef huart1;
UART_HandleTypeDef huart2;
GPIO_InitTypeDef GPIO_InitStruct;
osThreadId defaultTaskHandle;
osThreadId adc_sensorHandle;


/* Private function prototypes -----------------------------------------
*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
static void MX_USART1_UART_Init(void);
```

```
static void MX_USART2_UART_Init(void);
void StartDefaultTask(void const * argument);
void StartTask02(void const * argument);
void UART2_pins_config(void);
void UART1_pins_config(void);
void ADC_pins_config(void);



int main(void)
{

  /* -------MCU Configuration------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_ADC1_Init();
  MX_USART1_UART_Init();
  MX_USART2_UART_Init();
  UART1_pins_config();             //The same as HAL_UART_MspInit(huart1)
but speed setup as LOW
  UART2_pins_config();             //The same as HAL_UART_MspInit(huart2)
but speed setup as LOW
  HAL_ADC_MspInit(&hadc1);

  /* Create the thread(s) */
  /* definition and creation of defaultTask */
  osThreadDef(defaultTask, StartDefaultTask, osPriorityNormal, 0, 128);
  defaultTaskHandle = osThreadCreate(osThread(defaultTask), NULL);

  /* definition and creation of adc_sensor */
  osThreadDef(adc_sensor, StartTask02, osPriorityIdle, 1, 128);
  adc_sensorHandle = osThreadCreate(osThread(adc_sensor), NULL);


  /* Start scheduler */
  osKernelStart();

  while (1)
  {
  }

}

/** System Clock Configuration
*/
void SystemClock_Config(void)
```

```
{

  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;

    /**Configure the main internal regulator output voltage
    */
  __HAL_RCC_PWR_CLK_ENABLE();

  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);

    /**Initializes the CPU, AHB and APB busses clocks
    */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = 16;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Initializes the CPU, AHB and APB busses clocks
    */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Configure the Systick interrupt time
    */
  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

    /**Configure the Systick
    */
  HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

  /* SysTick_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(SysTick_IRQn, 15, 0);
}

/* ADC1 init function */
static void MX_ADC1_Init(void)
{

  ADC_ChannelConfTypeDef sConfig;
```

```
    /**Configure the global features of the ADC (Clock, Resolution, Data
Alignment and number of conversion)
    */
  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV2;
  hadc1.Init.Resolution = ADC_RESOLUTION_8B;
  hadc1.Init.ScanConvMode = DISABLE;
  hadc1.Init.ContinuousConvMode = DISABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DMAContinuousRequests = DISABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Configure for the selected ADC regular channel its corresponding
rank in the sequencer and its sample time.
    */
  sConfig.Channel = ADC_CHANNEL_0;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}

/* USART1 init function */
static void MX_USART1_UART_Init(void)
{

  huart1.Instance = USART1;
  huart1.Init.BaudRate = 9600;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}

/* USART2 init function */
static void MX_USART2_UART_Init(void)
```

```
{

  huart2.Instance = USART2;
  huart2.Init.BaudRate = 9600;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}

static void MX_GPIO_Init(void)
{

  GPIO_InitTypeDef GPIO_InitStruct;

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin : PC13 */
  GPIO_InitStruct.Pin = GPIO_PIN_13;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

}

void UART1_pins_config(void)
{
            __GPIOA_CLK_ENABLE();
            __USART1_CLK_ENABLE();

            /**USART1 GPIO Configuration
            PA9     ------> USART2_TX
            PA10     ------> USART2_RX
            */
            GPIO_InitStruct.Pin = GPIO_PIN_9|GPIO_PIN_10;
            GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
            GPIO_InitStruct.Pull = GPIO_PULLUP;
            GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
            GPIO_InitStruct.Alternate = GPIO_AF7_USART1;
            HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

}

void UART2_pins_config(void)
{
```

```
       //Using UART 2 to send data to PC

      __GPIOA_CLK_ENABLE();
      __USART2_CLK_ENABLE();

      /**USART2 GPIO Configuration
      PA2     ------> USART2_TX
      PA3     ------> USART2_RX
      */
      GPIO_InitStruct.Pin = GPIO_PIN_2|GPIO_PIN_3;
      GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;              //Alternate
Function Push Pull Mode
      GPIO_InitStruct.Pull = GPIO_PULLUP;
      GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
      GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
      HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

}

void ADC_pins_config(void)
{
      GPIO_InitTypeDef GPIO_InitStruct;
       __HAL_RCC_GPIOA_CLK_ENABLE();

     GPIO_InitStruct.Pin = GPIO_PIN_0;
     GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;              //Analog mode
     GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
     //GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

}


/* StartDefaultTask function */
void StartDefaultTask(void const * argument)
{

      /* Task : Read ADC value and send to PC through UART2 when pressed
number 4 */

    char *msg1 = "Analise de espasticidade\n\rLeitura GPIO A0:\n\r";
    char *msg2 = "Eletrogoniometro\n\r";
    float  adc_value_voltage = 0;
    char adc_value_voltage_char[10];
    float vec_data[4000];
    char vec_data_converted_char[10];
    float ang = 0;
    uint8_t vetor = 0;
    uint8_t n = 1;
    uint32_t nmax = 100;
    uint16_t adc_value = 0;
    uint32_t timeout = 1000;

    HAL_ADC_MspInit(&hadc1);   //clock and pins config
```

```
      while(1){

    osDelay(5);
    HAL_UART_Receive(&huart2, &vetor, 1, 10);
    osDelay(5);

    switch(vetor)
    {
    case '1':
      HAL_UART_Transmit(&huart2, msg1, strlen(msg1), 0xFFFF);
      vetor = 0;
      break;
    case '2':
      HAL_UART_Transmit(&huart2, msg2, strlen(msg2), 0xFFFF);
      vetor = 0;
      break;
    case '4':
      HAL_ADC_Start(&hadc1);
      while(HAL_ADC_PollForConversion(&hadc1, timeout ) != HAL_OK);
      adc_value = HAL_ADC_GetValue(&hadc1);
      HAL_ADC_Stop(&hadc1);

      adc_value_voltage = ((adc_value*3.3)/256); //register -> voltage
      ang = (adc_value_voltage*180)/3.3;             // voltage -> angle
      vec_data[n] = ang;                                        //write on
buffer

      float_to_string(adc_value_voltage, adc_value_voltage_char);
      float_to_string(vec_data[n], vec_data_converted_char);

      HAL_UART_Transmit(&huart2, "Tensao: ", strlen("Tensao: "), timeout);
      osDelay(1);
      HAL_UART_Transmit(&huart2, adc_value_voltage_char,
strlen(adc_value_voltage_char), timeout);
      adc_value = 0;
      vetor = 0;
      osDelay(1);
      HAL_UART_Transmit(&huart2, "\n\rAngulo correspondente: ",
strlen("\n\rAngulo correspondente: "), timeout);
      osDelay(1);
      HAL_UART_Transmit(&huart2, vec_data_converted_char,
strlen(vec_data_converted_char), timeout);
      osDelay(1);
      HAL_UART_Transmit(&huart2, "\n\r", strlen("\n\r"), timeout);
      n++;
      if(n == nmax){
            n=0;
      }

      break;
    case '5':

      break;
```

```
      }

    }

}


/* StartTask02 function */
void StartTask02(void const * argument)
{
      //Task: Read ADC1 and send to ESP8266 through UART1 continuously

    char angle_char[10];
    float adc_value_esp = 0, adc_value_voltage_esp = 0;
    uint32_t timeout = 1000;
    int nmax = 400, m = 5,t = 1;
    float buf[400];
    float ang_esp;
    float A1=2.3, A2=1.1, A0=1.4, sum=1;
      float tol = 0;
    float r1 = 0, r2 = 0,r2n = 0;
    char r1_char[10],r2_char[10], r2n_char[10];

       HAL_ADC_MspInit(&hadc1);   //clock and pins config

       /*read eletrogoniometer nmax samples*/
      for(int h=0; h<nmax; h++)
      {
            HAL_ADC_Start(&hadc1);
            while(HAL_ADC_PollForConversion(&hadc1, timeout ) != HAL_OK);
            adc_value_esp = HAL_ADC_GetValue(&hadc1);
            HAL_ADC_Stop(&hadc1);

          osDelay(5);

          adc_value_voltage_esp = ((adc_value_esp*3.3)/256);
      ang_esp = ((adc_value_voltage_esp*180)/3.3);
      buf[h] = ang_esp;

      float_to_string(buf[h], angle_char);

      HAL_UART_Transmit(&huart2, angle_char, strlen(angle_char), timeout);
//sendo to pc
      HAL_UART_Transmit(&huart2, "\n\r", strlen("\n\r"), timeout);
          HAL_UART_Transmit(&huart1, angle_char, strlen(angle_char),
timeout);   // send to nodemcu
            adc_value_esp = 0;


     }

/* sampling finished -> start the algorithm*/
```

```
      HAL_UART_Transmit(&huart2,"Fim da coleta\n\r", strlen("Fim da
coleta\n\r"), timeout);

    /********* recognition of first peak and first valley ************/

      for(int j=1; j<(nmax/2); j++){
            if(  (buf[j]>(buf[j+1]+tol)) && (buf[j]>(buf[j-1]+tol))   )
                  A1=buf[j];
      }

      for(int j=1; j<(nmax/2); j++){
                  if(  (buf[j]<(buf[j+1]-tol)) && (buf[j]<(buf[j-1]-tol))
)
                        A1=buf[j];
            }

      t = nmax-m;
      for(int k=t; k<nmax; k++){
            sum=sum + buf[k];
            }
             A0 = (1*sum)/(m+1);

            r1 = A1/((A1-A2)+0.01);          //avoid division by zero
            r2 = A1/A0;
            r2n = r2/1.6;

            float_to_string(r1, r1_char);
            HAL_UART_Transmit(&huart2,"r1", strlen("r1"), timeout);
            HAL_UART_Transmit(&huart2,r1_char, strlen(r1_char), timeout);

            float_to_string(r2, r2_char);
            HAL_UART_Transmit(&huart2,"r2", strlen("r2"), timeout);
            HAL_UART_Transmit(&huart2,r2_char, strlen(r2_char), timeout);

            float_to_string(r2n, r2n_char);
          HAL_UART_Transmit(&huart2,"rn", strlen("rn"), timeout);
          HAL_UART_Transmit(&huart2,r2n_char, strlen(r2n_char), timeout);

      /*************End of algorithm********/

      //loop of task
      while(1){

      }

}


void _Error_Handler(char * file, int line)
{
   while(1)
  {
  }
 }
```

```
#ifdef USE_FULL_ASSERT
void assert_failed(uint8_t* file, uint32_t line)
{

}

#endif

//*****************************End of code***************************
```

APÊNDICE B – CÓDIGO NODEMCU – MAC ADDRESS

```
/*************************************************************************
  * File Name          : macadress
  * Project            : Eletrogoniometro - Análise de espasticidade
  * Description        : Identify MAC Address
  * Author             : Pedro Henrique Pereira
  * Version            : 1
  * Year               : 2017
  *
 *************************************************************************/

#include <ESP8266WiFi.h>

void setup() {
  Serial.begin(115200);
  String clientMac = "";
  unsigned char mac[6];
  WiFi.macAddress(mac);
  clientMac += macToStr(mac);
  Serial.println();
  Serial.println(clientMac);
}
String macToStr(const uint8_t* mac){
 String result;
   for (int i = 0; i < 6; ++i) {
    result += String(mac[i], 16);
   if (i < 5)
    result += ':';
 }
 return result;
}

void loop() {
}
//*******************************End of code****************************
```

APÊNDICE C – CÓDIGO NODEMCU

```
/*************************************************************************
  * File Name          :nodemcu
  * Project            : Nodemcu Análise de espaticidade
  * Description        : Receive serial data and send to a broker using
PubSubClient library
  * Author             : Pedro Henrique Pereira
  * Version            : 1
  * Year               : 2017
  *
  * This code was implemented by the author using the pubsubclient library
under MIT License. Source: https://github.com/knolleary/pubsubclient
 *************************************************************************/

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <SoftwareSerial.h>

/*------------ WiFi login-------------------*/
const char* ssid = "x";
const char* password = "xx";

/*-----------Device authentication----------*/
#define ORG "quykgj"
#define DEVICE_TYPE "Nodemcu"
#define DEVICE_ID "5CCF7FB2EBF4"
#define TOKEN " - " //provided by IBM Watson IoT

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char topic[] = "iot-2/evt/status/fmt/json";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

WiFiClient wifiClient;
PubSubClient client(server, 1883, NULL, wifiClient);
SoftwareSerial mySerial(13, 15, false, 256);


/*----------Serial and WiFi Connections--------**/
void setup() {
 Serial.begin(115200);
 Serial.println("Connect! - Conexion COM5");

 mySerial.begin(9600);
```

```
 mySerial.println("Connect! - Conexion SOFTWARESERIAL COM9");


 Serial.print("Connecting to "); Serial.print(ssid);
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
 delay(500);
 Serial.print(".");
 }
 Serial.println("");

 Serial.print("WiFi connected, IP address: ");
Serial.println(WiFi.localIP());
}

/*------Variables----------*/
int n = 0;
int nmax;
int status_ = 0;
float r1=0,r2=0,r2n=0,data=0;
int flag_r1=0,flag_r2=0,flag_r2n = 0;
//float vec_data2[] = {1.9,2.7,3.4,4.3,5.2,6.6,7.4,8.7,9.8};
char string[32];
char byteRead;


/*----------Main task----------------*/
void loop() {

 if (!client.connected()) {
 Serial.print("Reconnecting client to ");
 Serial.println(server);
 while (!client.connect(clientId, authMethod, token)) {
 Serial.print(".");
 delay(500);
 }
 Serial.println();
 }

if(mySerial.available()){
  int num = mySerial.available();
  for(int i=0; i<num; i++)
  {
   string[i] = mySerial.read();
   Serial.write(string[i]);   //manual debug
  }
  Serial.write("\n\r");  //manual debug
```

```
  data=atof(string);
  if(flag_r1 == 1){
    r1=atof(string);
    flag_r1 =0;
  }

  if(flag_r2 == 1){
    r2=atof(string);
    flag_r2 =0;
  }

  if(flag_r2 == 1){
    r2=atof(string);
    flag_r2 =0;
  }

  if((string[0]=='r') && (string[1]=='1')){
    flag_r1 = 1;
  }

  if((string[0]=='r') && (string[1]=='2')){
    flag_r2n = 1;
  }

  if((string[0]=='r') && (string[1]=='n')){
    flag_r2 = 1;
  }
}

 /*--------Build payload-----------*/
 String payload = "{\"d\":{\"Name\":\"18FE34D81E46\"";
 payload += ",\"vec_data1\":";
 payload += data; //vec_data1[n];
 payload += ",\"R1\":";
 payload += r1;//r1;
 payload += ",\"R2\":";
 payload += r2;//r2;
 payload += ",\"R2n\":";
 payload += r2n;//r2n;
 payload += ",\"status\":";
 payload += status_;
 payload += ",\"interacao\":";
 payload += n++;
 payload += "}}";

 if(n==nmax){
  n=0;
```

```
 }

/*--------Sending payload-------------*/
 Serial.print("Sending payload: ");
 Serial.println(payload);

 if (client.publish(topic, (char*) payload.c_str())) {
 Serial.println("Publish ok");
 } else {
 Serial.println("Publish failed");
 }

 delay(500);
}
//*******************************End of code**************************
```