SÃO PAULO
FACULDADE SENAI DE TECNOLOGIA MECATRÔNICA
REVISTA BRASILEIRA DE MECATRÔNICA

SISTEMA PARA LEITURA DE VAZÃO DE ÁGUA *WIRELESS*

SYSTEM FOR READING OF WATER FLOW

Cassio da Silva e Souza[1], [i]
Douglas Airoldi[2], [ii]

RESUMO

Este artigo tem objetivo de obter a leitura de vazão de água e consumo, o sistema para obter os valores utiliza um sensor de fluxo instalado na entrada de uma residência, o microcontrolador é responsável pelo processamento e leitura dos dados. Após o processamento dos dados, os valores são enviados através do módulo *wireless* para um supervisório desenvolvido para *desktop*, um outro módulo *wireless* instalado no *desktop* recebe os valores enviados pelo microcontrolador onde podem ser visualizados em tempo real. Os resultados obtidos foram satisfatórios, a precisão do sistema está de acordo com a especificação pelo fabricante do sensor de fluxo, portanto para uso doméstico o sistema se mostra bastante útil para controlar o consumo e o uso consciente da água.

Palavras-chave: Vazão de água. Microcontrolador. Wireless. Supervisório.

ABSTRACT

This article has goal the reading of water flow and consumption, the system to obtain the values uses a flow sensor installed at the entrance of a residence, the microcontroller is responsible for processing and reading the data. After data processing, the values are sent through the wireless module to a desktop-developed supervisor, another wireless module installed on the desktop receives the values sent by the microcontroller where they can be viewed in real time. The results obtained were satisfactory, the accuracy of the system is in accordance with the manufacturer's specification of the flow sensor, so for domestic use the system proves to be very useful for controlling consumption and conscious use of water.

Keywords: Water flow. Microcontroller. Wireless. Supervisory.

Data de submissão: 07/06/2018
Data de aprovação: 25/09/2018

[1] Pós graduando em Automação Industrial. E-mail: cassio.1000@yahoo.com.br
[2] Professor Me da Faculdade SENAI de Tecnologia Mecatrônica. E-mail: douglas.airoldi@sp.senai.br

1 INTRODUÇÃO

Todas as atividades humanas dependem da água. Segundo a Sabesp (2017, p. 4) a: “Navegação, turismo, indústria, agricultura e geração de energia elétrica são alguns exemplos de seu uso econômico. O acesso à água é um direito humano fundamental”.

Com isso, o artigo traz uma proposta de monitorar esse consumo, conhecer em tempo real o quanto uma residência está consumindo água, compreender possíveis exageros e uso inconsciente da água.

Nesse artigo em específico, será demonstrado como coletar e armazenar os valores de vazão de água, a utilização e o tratamento dos dados, poderá ser explorada em um novo trabalho dando sequência a este com a leitura dos valores é possível desenvolver muitas aplicações. No supervisório desenvolvido para esse artigo, será mostrado a quantidade de litros de água consumido por uma residência.

O sensor de fluxo YF-S201, a placa de desenvolvimento com PIC 18F4550, dois módulos *wireless* HC-12 com um conversor USB -TTL RS232, são os componentes necessário para desenvolver essa aplicação.

2 DESENVOLVIMENTO

Para compor o *hardware*, foi utilizado um sensor de fluxo, uma placa de desenvolvimento com PIC 18F4550, dois módulos *wireless* HC-12 e um conversor USB -TTL RS232.

2.1 Sensor de fluxo YF-S201

O sensor de fluxo figura 1, tem o formato interno como uma turbina, com um imã acoplado e um sensor. Com a passagem da água, são gerados pulsos cuja frequência é diretamente proporcional ao volume de água.

Pinagem utilizada pelo sensor de fluxo:

a) vermelho: VCC;
b) preto: GND;
c) amarelo: Saída PWM.

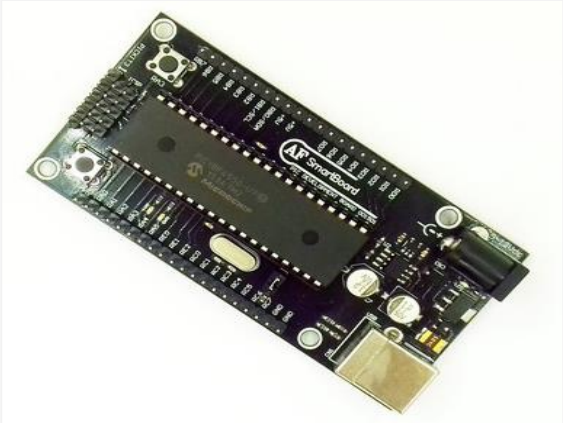
Figura 1 - Sensor de fluxo (YF-S201)

Fonte: Instituto Digital (2017)

2.2 Placa de desenvolvimento USB *AFSmartBoard*

A placa de desenvolvimento *AFSmartBoard* figura 2, contém um PIC18F4550, foi escolhida apenas para facilitar o projeto, logo não é necessário utilizar a mesma placa de desenvolvimento. Em outras aplicações, basta utilizar um microcontrolador que atenda as especificações.

Figura 2 - Placa de desenvolvimento USB AFSmartBoard



Fonte: AF Eletrônica (2017)

2.3 Módulo *wireless* HC-12

Com o módulo *wireless* HC-12 figura 3, é possível obter uma rede sem fio de até 1000m, e funciona com a comunicação serial.

O módulo *wireless* HC-12 utiliza rádio frequência para comunicação, a sua faixa de trabalho está entre 433.4 à 473MHz. Para configurar a faixa de trabalho e outros parâmetros é necessário utilizar o comando AT.

Figura 3 - Módulo *wireless* HC-12

Fonte: Arduino e Cia (2017)

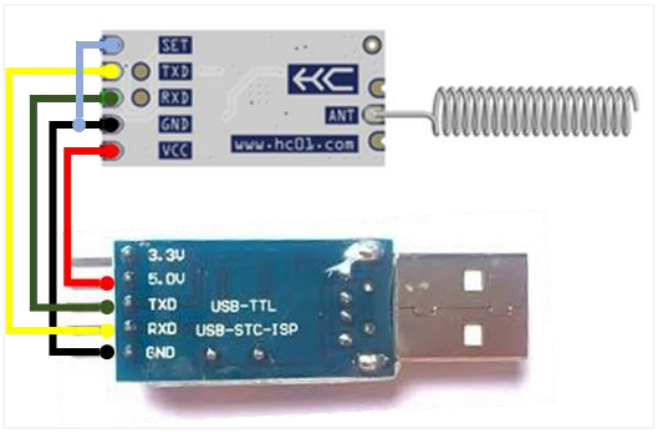
2.3.1 Configuração

Para configurar o módulo *wireless*, foi utilizado um conversor USB-TTL RS232. São necessários dois módulos para obter a comunicação, um fica instalado junto microcontrolador

e o outro fica conectado ao computador. Na figura 4 mostra a ligação entre os periféricos, ambos precisam ser configurados.

Para entrar no modo de configuração, linguagem de comando (AT), é necessário que o pino SET esteja ligado no GND.

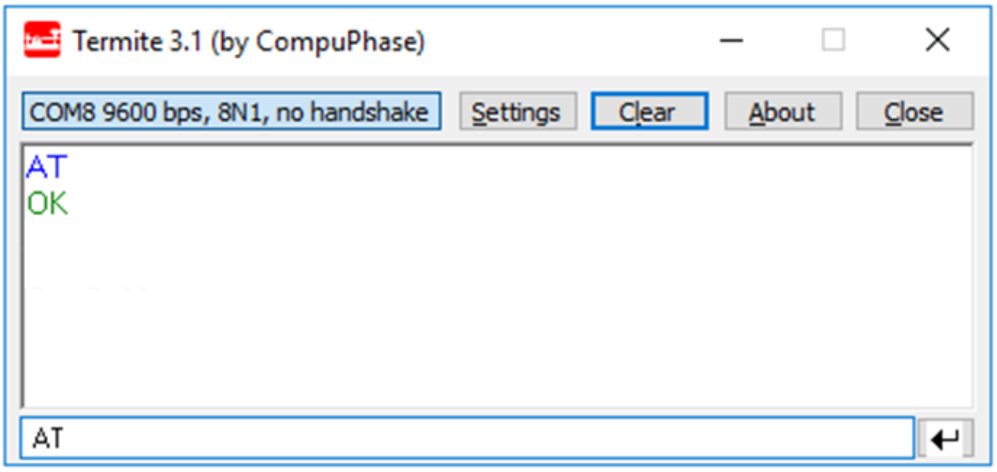
Figura 4 - Ligação entre os periféricos



Fonte: Elaborado pelo autor

Após realizar a montagem, é necessário um programa de terminal, para essa aplicação foi utilizado o Termite, configurado com velocidade padrão de 9600 bps. Para verificar se a comunicação foi realizada com sucesso, enviado o comando AT, o módulo deve responder OK, conforme figura 5.

Figura 5 – Uso do Termite



Fonte: Elaborado pelo autor

Com a comunicação realizada com sucesso, será necessário configurar os parâmetros de comunicação nos dois módulos, conforme o *datasheet* do HC-12 existe inúmeras configurações, para o projeto foi utilizada apenas os comandos:

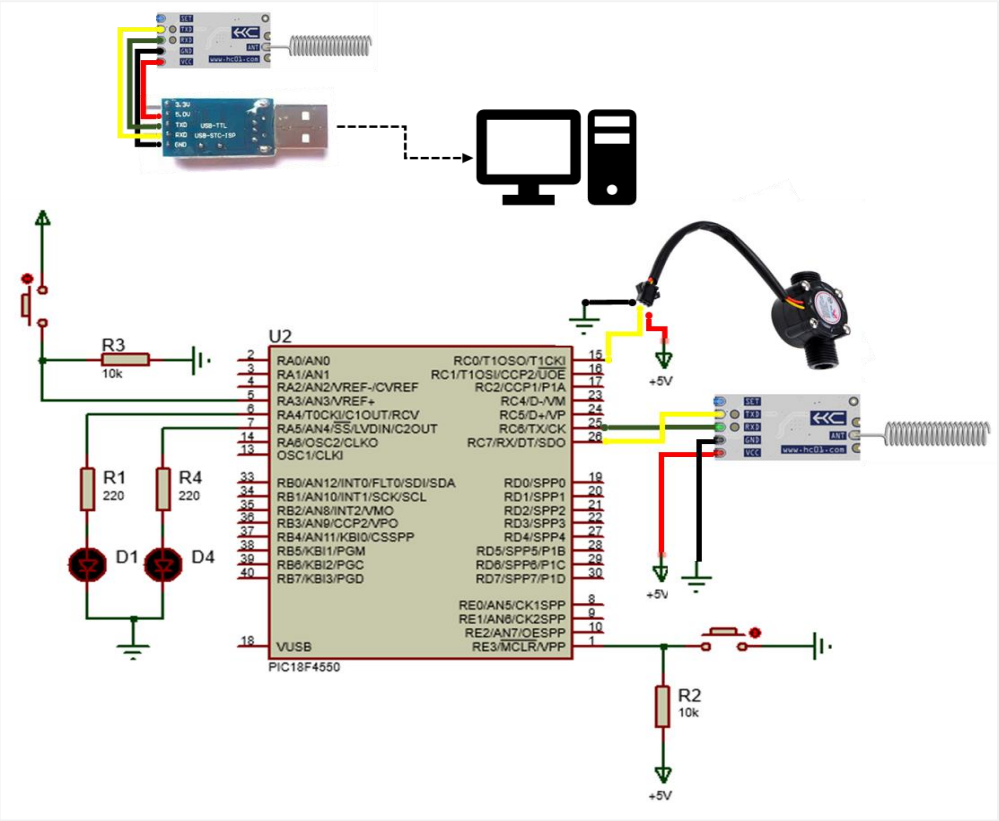
a) AT - Instrução de teste para retornar o comando OK;
b) AT+Bxxxx - Configuração da velocidade da porta serial, utilizado AT+B9600;
c) AT+Cxxx - Configuração do canal de comunicação *wireless*, onde aceita valores de 001 a 127, utilizado AT+C002;

d) AT+Px – Configuração da potência de transmissão, onde pode assumir valores de 1 a 8 necessário ler o *datasheet* para verificar as respectivas potências, utilizado AT+P1.

2.4 Sistema esquemático

A figura 6 a seguir apresenta o sistema esquemático:

Figura 6 - Sistema esquemático



Fonte: Elaborado pelo autor

3 SOFTWARE

O *software* para o microcontrolador foi desenvolvido no *mikroC PRO for PIC.* O supervisório foi desenvolvido no *VS Express for Desktop.*

3.1 Software MikroC for PIC

A seguir desenvolvimento do *software*.

```
//Programa: Sistema de leitura vazão de água
//Autor: Cassio Souza

//variáveis
unsigned cont = 0;
unsigned pulso = 0;
float vazao = 0;
```

```
float MiliLitros = 0;
float Litros =0;
unsigned long LitrosConv;
char txt_litros_minutos[15];
char txt_Litros[15];

void interrupt() //início da interrupção
{
   if (INTCON.TMR0IF)     //interrupção
   {
    cont++;           //armazenar a quantidade de interrupção
    RA4_bit =~ RA4_bit;  //LED indicar se a interrupçao está funcionando
   }
    INTCON.TMR0IF = 0;    //limpa interrupçao
    TMR0H        = 0x0B;
    TMR0L        = 0xDC;
} //fim da interrupçao

void main() {
    //config hardware
    RCON        = 0B10000000;
    ADCON1      = 0B00001111;
    CMCON       = 0B00000111;
    TRISC       = 1;
    PORTC       = 0;
    TRISA       = 0b11001111;
    PORTA       = 0;
    //Interrupção
    INTCON.GIE = 1;             //ativar interrupção global
    INTCON.PEIE = 1;            //ativar interrupçao por periférico
    INTCON.TMR0IE = 1;          //ativa interrupção timer 0
    //Timer0
    //Prescale 1:8; TMR0 Preload = 3036; Actual Interrupt Time: 100ms
    T0CON        = 0x82;
    TMR0H        = 0x0B;
    TMR0L        = 0xDC;
    //configuração serial
    UART1_Init(9600);           //Inicialização
    Delay_ms(100);              // Delay para estabilização
    //Timer 1 contar pulso
    T1CON = 0x03;               //habilita timer 1 e Clock externo
  while(1)
  {
    pulso = (TMR1H << 8 ) + TMR1L;   //contador de pulso RC0
    if(RA3_bit ==0) Litros = 0;      //zerar a variável litros
    if (cont == 5)                //leitura de litros a cada 1s
    {
     RA5_bit =~ RA5_bit;            //indicar a leitura
           vazao = pulso/5.5;            //fórmula da vazao, conforme              vazão=(float).(pulso)/5.5
     MiliLitros = vazao/60;         //conversão para mililitros
     Litros = (Litros + MiliLitros);  //conversão para litros
     LitrosConv =  Litros*1000;   //conversão para leitura no supervisório
     //necessário para leitura no supervisório
     LongWordToStrWithZeros(LitrosConv,txt_litros_minutos);
     intToStrWithZeros(vazao, txt_Litros);
     //protocolo de comunicação
     UART1_Write_Text("A");
     UART1_Write_Text(txt_litros_minutos);
     UART1_Write_Text("B");
```

```
    UART1_Write_Text(txt_Litros);

    cont = 0;
    TMR1H = 0;
    TMR1L = 0;
   }
  }
}
```

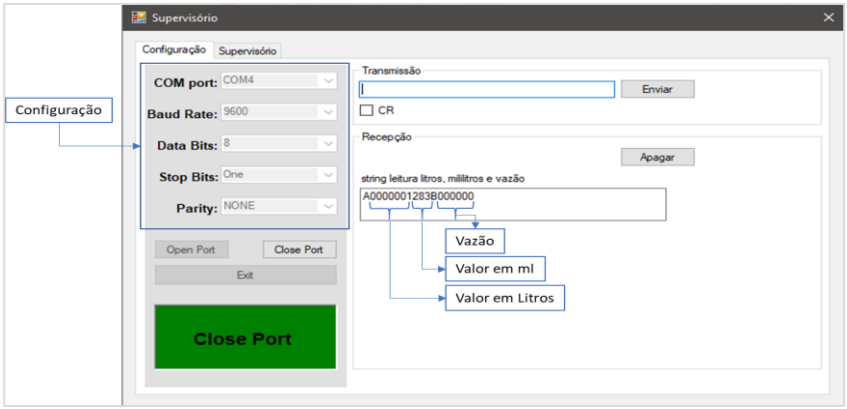
3.2 *Software VS Express for Desktop*

Utilizado para desenvolver o *software*, suporta a criação de aplicativos de *desktop* para Windows.

3.2.1 Tela inicial para configuração

Tela inicial de configuração e recepção dos dados. A figura 7 mostra a tela conectada e comunicando com o microcontrolador.

Figura 7 - Tela inicial configuração supervisório



Fonte: Elaborado pelo autor

3.2.2 Tela supervisório

A figura 8 mostra a tela principal, onde mostra o consumo de água e contém a figura de uma gota para indicar que o sistema está consumindo água ou não.

Figura 8 - Tela supervisório

Fonte: Elaborado pelo autor

3.2.3 Software

A seguir desenvolvimento do *software*.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;  //necessário para configurar a serial


namespace Supervisorio
{
   public partial class Form1 : Form
   {
      delegate void funcaoRecepcao(); //delegar alguma para função
      public Form1()
      {
         InitializeComponent();
         serialPort1.DataReceived += SerialPort1_DataReceived; //ler dados da seria += tab +tab para criar a função
      }
      private void SerialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e) //função criada
automaticamente para ler os dados da serial
      {
         funcaoRecepcao recepcaodelegate = new funcaoRecepcao(RecepcaoSerial); //Reservado memória para
recepção e cadastrando função de interrupção
         Invoke(recepcaodelegate);                              //será transferido para cá, depois envia para
RecepcaoSerial
      }
      string chtxt = null, str = null;
      public void RecepcaoSerial()
      {
         chtxt = serialPort1.ReadExisting();
         txtRecepcao.Text = chtxt;
         str = chtxt;
          chtxt = null;
         int numeroLitros;
         int numeroML;
         int numeroVazao;
         if (str.Substring(0, 1).Equals("A")) //tratando os valores
         {
           txtLitros.Text = str.Substring(8, 3);
           txtmlLitros.Text = str.Substring(1, 7);

            numeroLitros = Convert.ToInt32(txtmlLitros.Text);
            txtLitro_1.Text = numeroLitros.ToString();

            numeroML = Convert.ToInt32(txtLitros.Text);
            txtML.Text = numeroML.ToString();
         }
         if (str.Substring(11, 1).Equals("B"))
         {
            txtvazao.Text = str.Substring(12, 6);
```

```
        numeroVazao = Convert.ToInt32(txtvazao.Text);
        if (numeroVazao>1)
        {
            pictureBox2.Image = Supervisorio.Properties.Resources.Gota_ON1;
        }
        else
        {
            pictureBox2.Image = Supervisorio.Properties.Resources.Gota_OFF1;
        }
    }

}

private void btnExit_Click(object sender, EventArgs e)
{
    Close();
}

private void Form1_Load(object sender, EventArgs e)
{
    #region config_Port
    string[] ValoresPort = SerialPort.GetPortNames(); //recebe todas as portas seriais
    for (int i = 0; i < ValoresPort.Length; i++)
    {
        cbBoxPort.Items.Add(ValoresPort[i]); //inseri os nomes no cbBoxPort
    }
    cbBoxPort.Text = "COM1";
    #endregion
    #region Config_Baud
    int[] ValoresBaud = { 2400, 4800, 9600, 1920, 57600, 115200 };
    for (int i = 0; i < ValoresBaud.Length; i++)
    {
        cbBoxBaud.Items.Add(ValoresBaud[i].ToString());
    }
    cbBoxBaud.Text = "9600";
    #endregion
    #region Config_Data
    cbBoxData.Items.Add("7");
    cbBoxData.Items.Add("8");
    cbBoxData.Text = "8";
    #endregion
    #region Config_Stop
    cbBoxStop.Items.Add("None");
    cbBoxStop.Items.Add("One");
    cbBoxStop.Items.Add("Two");
    cbBoxStop.Text = "One";
    #endregion
    #region Config_Parity
    cbBoxParity.Items.Add("NONE");
    cbBoxParity.Items.Add("EVEN");
    cbBoxParity.Items.Add("ODD");
    cbBoxParity.Items.Add("MARK");
    cbBoxParity.Items.Add("SPACE");
    cbBoxParity.Text = "NONE";
    #endregion
}
private void btnCon_Click(object sender, EventArgs e)
{
    if (serialPort1.IsOpen == true) serialPort1.Close(); //
```

```
        else
        {
            serialPort1.PortName = cbBoxPort.Text;
            serialPort1.BaudRate = int.Parse(cbBoxBaud.Text);
            serialPort1.DataBits = int.Parse(cbBoxData.Text);
            serialPort1.StopBits = (StopBits)(cbBoxStop.SelectedIndex);
            serialPort1.Parity = (Parity)(cbBoxParity.SelectedIndex);
        }
        try //se não tiver erro
        {
            serialPort1.Open();
            btnCon.Enabled = false;
            btnDes.Enabled = true;
            btnExit.Enabled = false;
            cbBoxBaud.Enabled = false;
            cbBoxData.Enabled = false;
            cbBoxParity.Enabled = false;
            cbBoxPort.Enabled = false;
            cbBoxStop.Enabled = false;
            pnlMsg.BackColor = Color.Green;
            lblMsg.Text = "Close Port";
            pictureBox2.Image = Supervisorio.Properties.Resources.Gota_OFF1;
        }
        catch // se tiver erro
        {
            MessageBox.Show("Erro de comunicação com a porta!");
            btnCon.Enabled = true;
            btnDes.Enabled = false;
            btnExit.Enabled = true;
            cbBoxBaud.Enabled = true;
            cbBoxData.Enabled = true;
            cbBoxParity.Enabled = true;
            cbBoxPort.Enabled = true;
            cbBoxStop.Enabled = true;
            pnlMsg.BackColor = Color.Red;
            lblMsg.Text = "Open Port";
            pictureBox2.Image = Supervisorio.Properties.Resources.Gota_OFF1;
        }
    } //conectar

    private void btnDes_Click(object sender, EventArgs e)
    {
        try
        {
            serialPort1.Close();
            btnCon.Enabled = true;
            btnDes.Enabled = false;
            btnExit.Enabled = true;
            cbBoxBaud.Enabled = true;
            cbBoxData.Enabled = true;
            cbBoxParity.Enabled = true;
            cbBoxPort.Enabled = true;
            cbBoxStop.Enabled = true;
            pnlMsg.BackColor = Color.Red;
            lblMsg.Text = "Open Port";
            pictureBox2.Image = Supervisorio.Properties.Resources.Gota_OFF1;
        }
        catch
        {
```

```
            MessageBox.Show("Erro de comunicação com a porta!");
            btnCon.Enabled = false;
            btnDes.Enabled = true;
            btnExit.Enabled = false;
            cbBoxBaud.Enabled = false;
            cbBoxData.Enabled = false;
            cbBoxParity.Enabled = false;
            cbBoxPort.Enabled = false;
            cbBoxStop.Enabled = false;
            pnlMsg.BackColor = Color.Green;
            lblMsg.Text = "Close Port";
            pictureBox2.Image = Supervisorio.Properties.Resources.Gota_OFF1;
        }
    } //Desconectar

    private void btnEnviar_Click(object sender, EventArgs e) //enviar pela serial
    {
        if (serialPort1.IsOpen == true)
        {
            if (ChBoxEnviar.Checked)
            {
                serialPort1.Write(txtEnviar.Text + "\r");
            }
            else
            {
                serialPort1.Write(txtEnviar.Text);
            }
            txtEnviar.Text = "";
        }
        else
        {
            MessageBox.Show("Erro de comunicação com a porta!");
        }
    }

    private void btnApagar_Click(object sender, EventArgs e)
    {
        txtRecepcao.Text = "";
    }
    private void cbBoxPort_Click(object sender, EventArgs e) //carregar as portas sem precisar reiniciar o aplicativo
    {
        cbBoxPort.Items.Clear();
        string[] ValoresPort = SerialPort.GetPortNames();
        for (int i = 0; i < ValoresPort.Length; i++)
        {
            cbBoxPort.Items.Add(ValoresPort[i]);
        }

    }
  }
}
```

4 CONCLUSÃO

Os resultados obtidos após realização da montagem e testes do sistema, mostram-se aplicável para uma residência, onde pode se observar em tempo real o consumo de água, com a possibilidade de ampliar para aplicações mais especificas.

REFERÊNCIAS

AF ELETRÔNICA. **PIC 18F4550 AFSmart módulo de desenvolvimento**. 2017. Disponível em: <http://www.afeletronica.com.br/pd-296167-pic-18f4550-afsmart-modulo-de-desenvolvimento.html>. Acesso em: 06 out. 2017.

ARDUINO E CIA. **Comunicação sem fio com módulo wireless HC-12 e Arduino.** 2017. Disponível em: <http://www.arduinoecia.com.br/2016/11/modulo-wireless-hc-12-arduino.html>. Acesso em: 06 out. 2017.

INSTITUTO DIGITAL. **Sensor de Fluxo e Vazão de Água 1/2 - YF-S201**. 2017. Disponível em: <http://www.institutodigital.com.br/pd-24b6c2-sensor-de-fluxo-e-vazao-de-agua-1-2-yf-s201.html>. Acesso em: 06 out. 2017.

SABESP**.** Companhia de Saneamento Básico do Estado de São Paulo. **Manual de gerenciamento para controladores de consumo**. São Paulo: 2017**.** Disponível em: <http://site.sabesp.com.br/site/interna/Default.aspx?secaoId=340>. Acesso em: 06 out. 2017.

AGRADECIMENTOS

Primeiramente a Deus que é o maior mestre que alguém pode conhecer, que me permitiu acesso ao conhecimento ministrado durante todo o período do curso.

Agradeço aos professores por me proporcionar o conhecimento no processo de formação profissional.

A minha família e amigos que me ajudaram diretamente e indiretamente.

Sobre os autores:

[i] **CASSIO DA SILVA E SOUZA**

Possui graduação em Matemática pela Universidade UNIP (2010), possui graduação em Engenharia Mecânica pela Universidade Anhembi Morumbi (2015), cursando atualmente a [i]Pós-Graduação em Automação Industrial pela Faculdade SENAI de Tecnologia Mecatrônica (2017). Tem experiência na área de Engenharia de Processo. É Engenheiro de Processo na empresa MWM International responsável pelo processo de montagem de motores Diesel.

[ii] **DOUGLAS AIROLDI**

Possui graduação em Engenharia Elétrica com especialização em Eletrônica Plena Centro Universitário FEI (1993) e Mestrado (2011) pela Universidade Estadual de Campinas - UNICAMP. Atualmente é professor da Faculdade Senai de Tecnologia Mecatrônica, lecionando as disciplinas Análise de circuitos Elétricos, Eletrônica Analógica, Eletrônica Digital e Microcontroladores no curso Tecnológico em Mecatrônica e na Pós-Graduação em Automação Industrial. Tem experiência na área de Engenharia Elétrica, com ênfase em Automação Industrial, eletrônica embarcada, atuando principalmente nos seguintes temas: Mecatrônica, Automação Industrial.