Grupo 1:

Nomes: Gabriel de santi (202501001572) participou ativamente

Felipe Rocco (202301134315) participou ativamente

Ian Dias (202501000754) participou ativamente

Igor Costa (202502931931) participou ativamente

Lucas Callil (202401000371) participou ativamente

Rafael Correa (202502905408) participou ativamente

Kaue Fernandes (202502070527) participou ativamente

Relatório Técnico do Projeto: Sistema de Controle e Monitoramento com Arduino

Este documento detalha o desenvolvimento de um sistema de controle e monitoramento baseado na plataforma Arduino. O projeto integra múltiplos dispositivos de entrada (joystick analógico, teclado de membrana) e sensores (ultrassônico e de nível de água) para criar uma solução interativa e funcional, consolidando conceitos de arquitetura de computadores e sistemas embarcados.

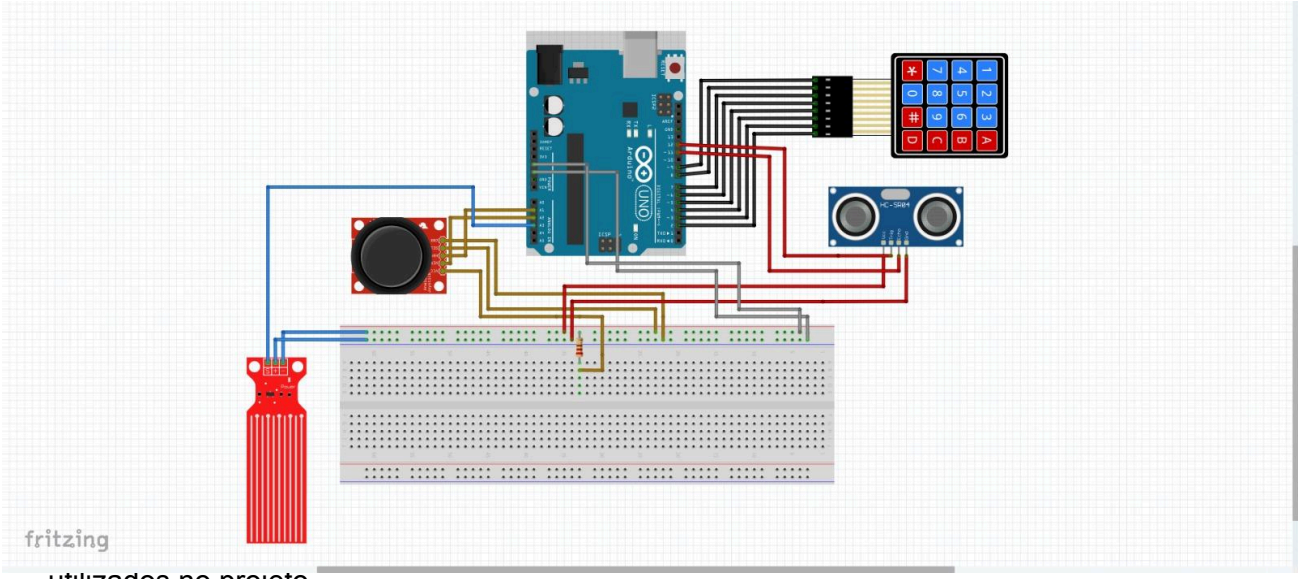
---

1. Descrição Textual do Funcionamento

O sistema foi projetado para operar de forma modular, permitindo ao usuário escolher diferentes modos de monitoramento e configurar parâmetros de alerta em tempo real. O fluxo de operação é o seguinte:

1. **Inicialização:** Ao ser ligado, o sistema exibe uma mensagem de boas-vindas no monitor serial e aguarda o comando do usuário.
2. **Seleção de Modo via Joystick:** O usuário utiliza o joystick analógico para pré-selecionar um dos três modos de operação:
    - **Joystick para a Esquerda:** Pré-seleciona o **Modo de Monitoramento Contínuo**.
    - **Joystick para a Direita:** Pré-seleciona o **Modo de Monitoramento Sob Demanda**.
    - **Joystick para Cima:** Pré-seleciona o **Modo de Teste de Sensores**.

3. **Confirmação via Teclado:** Após mover o joystick, a escolha fica pendente. O usuário deve confirmá-la pressionando a tecla # no teclado de membrana. A tecla * cancela a seleção.
4. **Execução dos Modos:**
    - **Modo Contínuo:** O sistema lê os sensores de distância e de nível de água a cada 1 segundo e exibe os resultados no monitor serial.
    - **Modo Sob Demanda:** O sistema permanece em espera. A leitura e exibição dos dados dos sensores ocorrem apenas quando o usuário pressiona a tecla *.
    - **Modo de Teste:** O sistema realiza uma única leitura de cada sensor, exibe seu status e os valores lidos, e retorna ao estado inicial de seleção de modo.
5. **Parametrização de Alertas:** A qualquer momento, o usuário pode usar o teclado para configurar os limites de alerta:
    - **Tecla '1'**: Inicia o processo para definir a **distância crítica** (em cm). O usuário digita o valor e confirma com #.
    - **Tecla '2'**: Inicia o processo para definir o **nível de água crítico** (em %). O usuário digita o valor e confirma com #.



utilizados no projeto.

3. Fluxograma Simplificado do Sistema

3.1. Resumo do Funcionamento

O sistema lê comandos de um teclado e de um joystick. Com base nas entradas, ele muda o modo de operação e realiza leituras dos sensores de distância e nível de água. Os dados são exibidos no monitor serial, e o sistema gera alertas quando os valores ultrapassam os limites configurados.

3.2. Estrutura Lógica do Sistema

INÍCIO DO SISTEMA

→ Arduino liga e prepara tudo (setup)

→ Entra no loop principal

LEITURA DOS CONTROLES

→ Lê o teclado

→ Lê o joystick

MOVIMENTO DO JOYSTICK

→ Esquerda: modo contínuo

→ Direita: modo sob demanda

→ Cima: modo teste de sensores

CONFIRMAÇÃO PELO TECLADO

→ '#' confirma o modo

→ '*' cancela

→ Outras teclas são ignoradas

AJUSTE DE PARÂMETROS

→ '1' define distância crítica

→ '2' define nível de água crítico

→ '#' confirma valor

→ '*' cancela ajuste

EXECUÇÃO DOS MODOS

→ Contínuo: leitura automática a cada 1 segundo

→ Sob Demanda: leitura apenas ao pressionar '*'

→ Teste: leitura única e retorno ao início

LEITURA DE SENSORES

→ Sensor ultrassônico: mede distância (cm)

→ Sensor analógico: mede nível de água (%)

EXIBIÇÃO DOS RESULTADOS

→ Mostra valores no monitor serial

→ Se distância < limite → alerta de proximidade

→ Se nível > limite → alerta de nível alto

RETORNO

→ Volta ao loop principal 3. Etapas Principais

3.3. Etapas principais

Etapa	Ação Principal	Entrada	Saída
1	Início e configuração	-	Sistema pronto
2	Leitura de joystick	Movimento	Escolha do modo
3	Confirmação via teclado	# / *	Modo confirmado ou cancelado
4	Ajuste de parâmetros	Teclas numéricas	Novos limites definidos
5	Leitura de sensores	Sinais analógicos	Valores de distância e nível
6	Exibição de resultados	Dados lidos	Mostra valores e alertas
7	Retorno ao loop	-	Recomeça o ciclo

4. Exemplos de Entrada e Saídas (Monitor Serial)

A seguir, exemplos de interação do usuário com o sistema e as respectivas saídas exibidas no monitor serial.

Cenário 1: Inicialização do Sistema

INICIADO: Joystick + Teclado + Sensores ---

TECLADO: '#' confirma | '*' cancela/limpa

'1' setar Distancia Critica (cm)

'2' setar Nivel de Agua Critico (%)

Use o JOYSTICK para escolher modo e confirme no teclado ('#' confirma, '*' cancela).

Ou pressione '1'/'2' para configurar parametros.

Cenário 2: Seleção do Modo Contínuo e Leitura

O usuário move o joystick para a esquerda e depois pressiona '#' no teclado.

>> Selecao pendente: 1 - MONITORAMENTO CONTINUO

Pressione '#' para CONFIRMAR ou '*' para CANCELAR no teclado.

*** MODO SELECIONADO: 1 - MONITORAMENTO CONTINUO *

Distancia (cm): 125 | Nivel Agua (%): 15

Distancia (cm): 124 | Nivel Agua (%): 15

Cenário 3: Configuração de Parâmetro e Geração de Alerta

O usuário pressiona '1', digita '30' e confirma com '#'. Na leitura seguinte, o objeto está a 25 cm.

Digite a DISTANCIA CRITICA em cm e pressione '#'. Use '*' para cancelar.

30

Parametro atualizado: distancia critica = 30 cm.

Distancia (cm): 25 | Nivel Agua (%): 18

ALERTA: distancia inferior ao limite!

Cenário 4: Seleção e Uso do Modo Sob Demanda

O usuário move o joystick para a direita, confirma com '#'. Depois de um tempo, pressiona ".

>> Selecao pendente: 2 - MONITORAMENTO SOB DEMANDA

Pressione '#' para CONFIRMAR ou '*' para CANCELAR no teclado.

*** MODO SELECIONADO: 2 - MONITORAMENTO SOB DEMANDA *

No MODO SOB DEMANDA: pressione '*' para ler sensores agora.

[LEITURA SOB DEMANDA ACIONADA (*)]

Distancia (cm): 210 | Nivel Agua (%): 45

5. Código Fonte Completo

Abaixo, o código-fonte completo que implementa toda a lógica descrita.

—---

```
//  Controle por Joystick + Teclado de Membrana 4x4
//  Modos: Contínuo, Sob Demanda e Teste de Sensores
//  Parametrização via teclado: distância crítica (cm) e
//  nível de água crítico (%)
//  Autor: Gabriel De Santi Jorge
// ============================================================


#include <Keypad.h>


// -------------------------
// Teclado 4x4: linhas (9,8,7,6) | colunas (5,4,3,2)
// -------------------------
const byte ROWS = 4, COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
```

```
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

// ========================================================
// TIPOS / VARIÁVEIS GLOBAIS
// ========================================================
struct LeituraSensores {
  long distanciaCM;   // cm (ultrassônico)
  int  nivelPct;      // % (0–100)
};

LeituraSensores ultimaLeitura;   // global com a última leitura

// Joystick
const int pinoEixoX = A1;
const int pinoEixoY = A2;

// Zona morta do joystick
const int CENTER_MIN = 400;
const int CENTER_MAX = 600;

// Ultrassônico HC-SR04
```

```
const int trigPin = 11;
const int echoPin = 12;

// Nível de água (A0 → 0..100%)
const int waterPin = A0;

// Modos
#define MODE_INITIAL        0
#define MODE_CONTINUOUS     1
#define MODE_ONDEMAND       2
#define MODE_SENSOR_TEST    3

int currentMode = MODE_INITIAL;
int pendingMode = MODE_INITIAL;
bool waitingConfirmation = false;

// Limites configuráveis
int limiteDistanciaCM = 50;   // cm
int limiteNivelPct    = 70;   // %

const int DELAY_CONTINUOUS_MS = 1000;

// Entrada via teclado
String inputBuffer = "";
enum InputTarget { NONE, DISTANCIA_CM, NIVEL_PCT };
InputTarget inputDestino = NONE;
```

```
// ========================================================

// FUNÇÕES DO SISTEMA

// ========================================================


void printLinhaSeparador() { Serial.println("----------------------------------------"); }


void printAjudaTecladoBase() {

  Serial.println("TECLADO: '#' confirma | '*' cancela/limpa");

  Serial.println("         '1' setar Distancia Critica (cm)");

  Serial.println("         '2' setar Nivel de Agua Critico (%)");

}


void printModoSelecionado(const char* nome) {

  Serial.print("\n*** MODO SELECIONADO: ");

  Serial.print(nome);

  Serial.println(" ***");

}


void printSelecaoPendente(int selMode) {

  Serial.println();

  Serial.print(">> Selecao pendente: ");

  if (selMode == MODE_CONTINUOUS) Serial.println("1 - MONITORAMENTO
CONTINUO");

  else if (selMode == MODE_ONDEMAND) Serial.println("2 - MONITORAMENTO SOB
DEMANDA");

  else if (selMode == MODE_SENSOR_TEST) Serial.println("3 - TESTE DE SENSORES");
```

```
  Serial.println("Pressione '#' para CONFIRMAR ou '*' para CANCELAR no teclado.");
}

void requestModeConfirmation(int selMode) {
  pendingMode = selMode;
  waitingConfirmation = true;
  printSelecaoPendente(selMode);
}

// --- Funções de Leitura de Sensores ---
long readDistanceCm() {
  digitalWrite(trigPin, LOW); delayMicroseconds(2);
  digitalWrite(trigPin, HIGH); delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH, 30000UL); // timeout ~30ms
  if (duration == 0) return -1;
  return duration / 58; // cm
}

int readWaterLevelPct() {
  int raw = analogRead(waterPin);
  return map(constrain(raw, 0, 1023), 0, 1023, 0, 100);
}

void leSensores() {
  ultimaLeitura.distanciaCM = readDistanceCm();
```

```
  ultimaLeitura.nivelPct    = readWaterLevelPct();
}

// --- Exibição e Alertas ---
void exibeResultados(bool mostrarCabecalho) {
  if (mostrarCabecalho) printLinhaSeparador();
  Serial.print("Distancia (cm): ");
  Serial.print(ultimaLeitura.distanciaCM);
  Serial.print(" | Nivel Agua (%): ");
  Serial.println(ultimaLeitura.nivelPct);

  if (ultimaLeitura.distanciaCM >= 0 && ultimaLeitura.distanciaCM < limiteDistanciaCM) {
    Serial.println("ALERTA: distancia inferior ao limite!");
  }
  if (ultimaLeitura.nivelPct > limiteNivelPct) {
    Serial.println("ALERTA: nivel de agua critico!");
  }
}

// --- Leitura de Entradas ---
void leJoystick() {
  if (waitingConfirmation || inputDestino != NONE) return;

  int x = analogRead(pinoEixoX);
  int y = analogRead(pinoEixoY);
```

```
  if (x < CENTER_MIN)          requestModeConfirmation(MODE_CONTINUOUS);
  else if (x > CENTER_MAX)     requestModeConfirmation(MODE_ONDEMAND);
  else if (y < CENTER_MIN)     requestModeConfirmation(MODE_SENSOR_TEST);
}

// --- Lógica do Teclado ---
// (dividida em várias funções auxiliares para clareza)

void finalizarEntradaParametro() {
    if (inputDestino == NONE) return;

    if (inputBuffer.length() == 0) {
        Serial.println("Entrada vazia. Operacao cancelada.");
        inputDestino = NONE;
        return;
    }

    long valor = inputBuffer.toInt();
    if (inputDestino == DISTANCIA_CM) {
        if (valor <= 0 || valor > 400) {
            Serial.println("Valor invalido (use 1..400 cm). Operacao cancelada.");
        } else {
            limiteDistanciaCM = (int)valor;
            Serial.print("Parametro atualizado: distancia critica = ");
            Serial.print(limiteDistanciaCM);
            Serial.println(" cm.");
```

```
        }
    } else if (inputDestino == NIVEL_PCT) {
        if (valor < 0 || valor > 100) {
            Serial.println("Valor invalido (0..100%). Operacao cancelada.");
        } else {
            limiteNivelPct = (int)valor;
            Serial.print("Parametro atualizado: nivel de agua critico = ");
            Serial.print(limiteNivelPct);
            Serial.println(" %.");
        }
    }

    inputDestino = NONE;
    inputBuffer  = "";
}

void cancelarEntradaParametro() {
    inputDestino = NONE;
    inputBuffer  = "";
    Serial.println("Entrada cancelada.");
}

void promptEntradaParametro() {
    if (inputDestino == DISTANCIA_CM)
        Serial.println("Digite a DISTANCIA CRITICA em cm e pressione '#'. Use '*' para
cancelar.");
    else if (inputDestino == NIVEL_PCT)
```

```
      Serial.println("Digite o NIVEL DE AGUA CRITICO em % e pressione '#'. Use '*' para cancelar.");

}


void leTeclado() {

   char key = keypad.getKey();

   if (!key) return;


   // Se estiver aguardando confirmação de modo

   if (waitingConfirmation) {

      if (key == '#') {

         waitingConfirmation = false;

         if     (pendingMode == MODE_CONTINUOUS) { currentMode = MODE_CONTINUOUS; printModoSelecionado("1 - MONITORAMENTO CONTINUO"); }

         else if (pendingMode == MODE_ONDEMAND)   { currentMode = MODE_ONDEMAND;   printModoSelecionado("2 - MONITORAMENTO SOB DEMANDA"); Serial.println("No MODO SOB DEMANDA: pressione '*' para ler sensores agora."); }

         else if (pendingMode == MODE_SENSOR_TEST){ currentMode = MODE_SENSOR_TEST;printModoSelecionado("3 - TESTE DE SENSORES"); }

      } else if (key == '*') {

         waitingConfirmation = false;

         pendingMode = MODE_INITIAL;

         Serial.println("Selecao cancelada.");

      }

      return;

   }


   // Se estiver no meio da entrada de um parâmetro

   if (inputDestino != NONE) {
```

```
        if (key >= '0' && key <= '9') {
            inputBuffer += key; Serial.print(key);
        } else if (key == '#') {
            Serial.println();
            finalizarEntradaParametro();
        } else if (key == '*') {
            Serial.println();
            cancelarEntradaParametro();
        }
        return;
    }

    // Comandos principais
    if (key == '1') {
        inputDestino = DISTANCIA_CM; inputBuffer = ""; promptEntradaParametro();
    } else if (key == '2') {
        inputDestino = NIVEL_PCT;     inputBuffer = ""; promptEntradaParametro();
    } else if (key == '*') {
        if (currentMode == MODE_ONDEMAND) {
            Serial.println("\n[LEITURA SOB DEMANDA ACIONADA (*)]");
            leSensores();
            exibeResultados(true);
        }
    }
}
```

```
// ========================================================

// SETUP E LOOP PRINCIPAL

// ========================================================

void setup() {

  Serial.begin(9600);


  pinMode(trigPin, OUTPUT);

  pinMode(echoPin, INPUT);


  printLinhaSeparador();

  Serial.println("--- INICIADO: Joystick + Teclado + Sensores ---");

  printAjudaTecladoBase();

  Serial.println("Use o JOYSTICK para escolher modo e confirme no teclado ('#' confirma, '*'
cancela).");

  Serial.println("Ou pressione '1'/'2' para configurar parametros.");

  printLinhaSeparador();

}


void loop() {

  leTeclado();

  leJoystick();


  switch (currentMode) {

    case MODE_CONTINUOUS:

      leSensores();

      exibeResultados(true);

      delay(DELAY_CONTINUOUS_MS);
```

```
  break;

case MODE_ONDEMAND:
  // A leitura ocorre na função leTeclado() ao pressionar '*'
  delay(10);
  break;

case MODE_SENSOR_TEST:
  printLinhaSeparador();
  leSensores();
  if (ultimaLeitura.distanciaCM >= 0) {
    Serial.print("Sensor de distancia OK – leitura atual: ");
    Serial.print(ultimaLeitura.distanciaCM);
    Serial.println(" cm");
  } else {
    Serial.println("Sensor de distancia sem leitura (timeout).");
  }
  Serial.print("Sensor de nivel de agua OK – leitura atual: ");
  Serial.print(ultimaLeitura.nivelPct);
  Serial.println(" %");
  printLinhaSeparador();

  currentMode = MODE_INITIAL; // Retorna ao modo inicial após o teste
  break;

default: // MODE_INITIAL
```

```
        delay(20);
        break;
    }
}
```