Funcionamento

A aplicação exemplo executa a leitura de duas entradas digitais da Shield Bravas e escreve nas saídas a relé da seguinte forma:

- Entrada digital 1 ativa: Reles são acionados, um a um, do relé 1 ao relé 8, com intervalo entre acionamentos de 1s.
- Entrada digital 2 ativa: Reles são acionados, um a um, do relé 8 ao relé 1, com intervalo entre acionamentos de 1s.
- Entradas digitais 1 e 2 acionadas simultaneamente, aplicativo é encerrado.

Um breve “menu” é apresentado na porta serial do equipamento mostrando as informações acima.

Ligações elétricas

Saídas

A placa Shield Bravas possui oito relés com saída reversível. O acionamento dos relés é indicado por LEDs montados na própria placa. Na aplicação exemplo podemos verificar o acionamento dos relés através dos LEDs.

Entradas

As entradas digitais da placa Shield Bravas são opto-acopladas. Para acionar essas entradas podemos usar a saída 12Vdc existente na própria placa fechando o circuito com micro-chaves ou botões para as entradas, conforme diagrama abaixo:

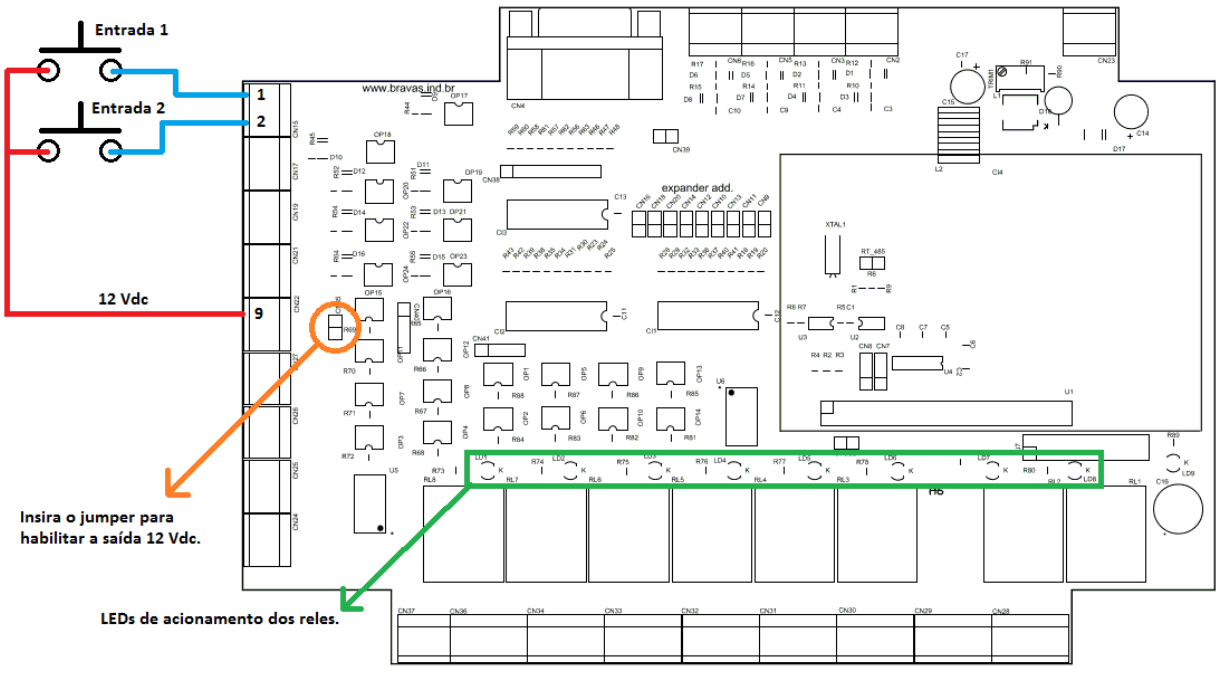

Fig. 1

As saídas e entradas digitais da Shield Bravas são conectadas a Raspberry Pi através do canal I²C, usando o circuito integrado PCF8574.

GPIO Library

A aplicação exemplo faz uso de uma biblioteca open source chamada wiringpi, em especial a extensão PCF8574 para ler e escrever através do barramento i2c.

Para ler as entradas digitais usamos:

int digitalRead (int pin) ;

Para escrever nas saidas digitais, usamos:

void digitalWrite (int pin, int value) ;

Os valores para os pinos (pin), no caso da comunicação i2c com a wiringpi, são de definidos durante a inicialização da biblioteca.

Mais informações sobre a biblioteca podem ser encontradas nos links abaixo:

Biblioteca:
http://wiringpi.com

How to install:
http://wiringpi.com/download-and-install/

PCF8574 Extension:
http://wiringpi.com/extensions/i2c-pcf8574/

Sample code:

main.cc

```
/*****************************************************************************/
/**
 * \file
 * \author  Bravas Tecnologia
 * \brief   Main
 *
 * Purpose: Sample application for Bravas Shield.
 */
/*****************************************************************************/

/*****************************************************************************
 *
 *   (C) Copyright Bravas Tecnologia - 2017.  All rights reserved.
 *
 *****************************************************************************/

#include <stdio.h>
#include <string.h>
#include <iostream>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#include <signal.h>
#include <wiringPi.h>
#include <pcf8574.h>

/* Project includes */
#include "defines.h"

/**
 * \brief Initiate board hardware
 * \param none
 */
void hal_init();

/**
 * \brief Update control variable with input status
 * \param none
 */
void refresh_inputs();

/**
 * \brief Update outputs status
 * \param none
```

```
 */
void refresh_outputs();
bool exit_main_loop;
bool input1_on;
bool input2_on;

/*****************************************************************************/
int main(int argc, char *argv[])
{
   // Setup Shield hardware
   hal_init();

   // Simple info...
   std::cout << std::endl;
   std::cout << std::endl;
   std::cout << std::endl;
   std::cout << "\t**************************************************" << std::endl;
   std::cout << "\t**********   Bravas Tecnologia  **********" << std::endl;
   std::cout << "\t**************************************************" << std::endl;
   std::cout << std::endl;
   std::cout << "\tAcione a Entrada 1 para rotacionar a direta"<< std::endl;
   std::cout << "\tAcione a Entrada 2 para rotacionar a esquera"<< std::endl;
   std::cout << "\tAcione as Entrada 1 e 2 ao mesmo tempo para sair da aplicação"<< std::endl;

   // Init vars
   exit_main_loop = false;
   input1_on = false;
   input2_on = false;

   // Main loop
   while (!exit_main_loop)
   {
      // Read inputs
      refresh_inputs();

      // Relay flow
      refresh_outputs();
   }

   std::cout << "\tBye bye!!"<< std::endl;
   return 0;
}

/*****************************************************************************/
void hal_init()
{
   // Init drivers
   wiringPiSetup();
```

```
    pcf8574Setup(100, 0x25);    // Relay outputs
    pcf8574Setup(116, 0x26);    // Open Collector outputs
    pcf8574Setup(132, 0x27);    // Input

    // Setup GPIO direction
    for (unsigned int i = 0; i < sizeof(rasp_gpio) / sizeof(rasp_gpio[0]); ++i)
    {
        pinMode (rasp_gpio[i].pin_, rasp_gpio[i].dir_);
        if (rasp_gpio[i].dir_ == OUTPUT)
        {
            // Start outputs low as default
            digitalWrite (rasp_gpio[i].pin_, HIGH);
        }
    }
}

/*****************************************************************************/
void refresh_inputs()
{
    // This is a simple debounce routine...do not use in production
    uint32_t cur_st_1, lst_st_1;
    uint32_t cur_st_2, lst_st_2;

    // Read from hardware
    cur_st_1 = digitalRead(INPUT1);
    cur_st_2 = digitalRead(INPUT2);

    // Sleep and check again
    usleep(250000);

    // Read from hardware again
    lst_st_1 = digitalRead(INPUT1);
    lst_st_2 = digitalRead(INPUT2);

    if (cur_st_1 == lst_st_1)
    {
        if (lst_st_1 == HIGH)
        {
            // just pressed
            input1_on = true;
        }
        else
        {
            // just pressed
            input1_on = false;
        }
    }
```

```
   if (cur_st_2 == lst_st_2)
   {
      if (lst_st_2 == HIGH)
      {
         // just pressed
         input2_on = true;
      }
      else
      {
         // just pressed
         input2_on = false;
      }
   }

}

/*****************************************************************************/
void refresh_outputs()
{
   static bool clockwise;
   static uint32_t curId;
   static uint32_t lstId;
   static uint32_t delay;

   // Simple delay to update every 1s (the input loops takes about 250ms
   if (delay++ < 4)
      return;
   delay = 0;

   // Check for exit request
   if (input1_on && input2_on)
   {
      exit_main_loop = true;
      return;
   }

   // No button
   if (!input1_on && !input2_on)
   {
      return;
   }

   // Input 1
   if (input1_on)
   {
      if (lstId >= 7) {
         curId = 0;
      }
```

```
      else {
         curId = lstId + 1;
      }
   }
   else
      if (input2_on) // Input 2
      {
         if (lstId <= 0) {
            curId = 7;
         }
         else {
            curId = lstId - 1;
         }
      }

   // Turn on one relay an turn off the old one
   digitalWrite (rasp_gpio[curId].pin_, LOW);
   digitalWrite (rasp_gpio[lstId].pin_, HIGH);
   lstId = curId;
}
```

define.h

```
/*****************************************************************************/
/**
 * \file
 * \author  Bravas Tecnologia
 * \brief   General definitiions
 */
/*****************************************************************************/

/*****************************************************************************
 *
 *   (C) Copyright Bravas Sistemas Ltda - 2016.  All rights reserved.
 *
 *****************************************************************************/

#ifndef DEFINES_H_
#define DEFINES_H_

/*****************************************************************************/
/*                                  Includes                                 */
/*****************************************************************************/
#include <stdint.h>

#define FAILED              -1
```

```
#define SUCCESS             0

/* Hardware expander i2c addresses */
#define    OUTPUT_RELAY_ADD   100
#define    OUTPUT_TRANS_ADD   116
#define    INPUT_ADD          132

#define    INPUT1             INPUT_ADD+0
#define    INPUT2             INPUT_ADD+1

static struct st_rasp_gpio_ {
   uint32_t   id_;
   uint32_t   pin_;
   uint32_t   dir_;
   uint32_t   type;
} rasp_gpio[]={
      {0,  OUTPUT_RELAY_ADD + 0, OUTPUT },
      {1,  OUTPUT_RELAY_ADD + 1, OUTPUT },
      {2,  OUTPUT_RELAY_ADD + 2, OUTPUT },
      {3,  OUTPUT_RELAY_ADD + 3, OUTPUT },
      {4,  OUTPUT_RELAY_ADD + 4, OUTPUT },
      {5,  OUTPUT_RELAY_ADD + 5, OUTPUT },
      {6,  OUTPUT_RELAY_ADD + 6, OUTPUT },
      {7,  OUTPUT_RELAY_ADD + 7, OUTPUT },
      {8,  OUTPUT_TRANS_ADD + 0, OUTPUT },
      {9,  OUTPUT_TRANS_ADD + 1, OUTPUT },
      {10, OUTPUT_TRANS_ADD + 2, OUTPUT },
      {11, OUTPUT_TRANS_ADD + 3, OUTPUT },
      {12, OUTPUT_TRANS_ADD + 4, OUTPUT },
      {13, OUTPUT_TRANS_ADD + 5, OUTPUT },
      {14, OUTPUT_TRANS_ADD + 6, OUTPUT },
      {15, OUTPUT_TRANS_ADD + 7, OUTPUT },
      {16, INPUT_ADD + 0      , INPUT  },
      {17, INPUT_ADD + 1      , INPUT  },
      {18, INPUT_ADD + 2      , INPUT  },
      {19, INPUT_ADD + 3      , INPUT  },
      {20, INPUT_ADD + 4      , INPUT  },
      {21, INPUT_ADD + 5      , INPUT  },
      {22, INPUT_ADD + 6      , INPUT  },
      {23, INPUT_ADD + 7      , INPUT  },
      {24, 7                  , INPUT  },
      {25, 2                  , INPUT  },
      {26, 3                  , INPUT  },
      {27, 12                 , INPUT  },
};

#endif /* DEFINES_H */
```