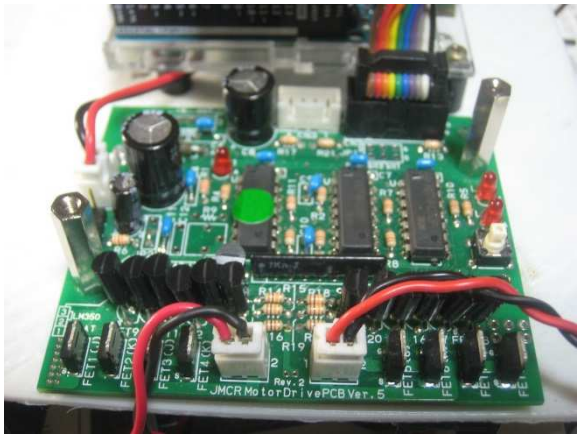
Logicool F710Wireless Gamepad を用いた搬送車製作

Cytron の HP に掲載されている搬送車を Arduino UNO で作り変えてみました。

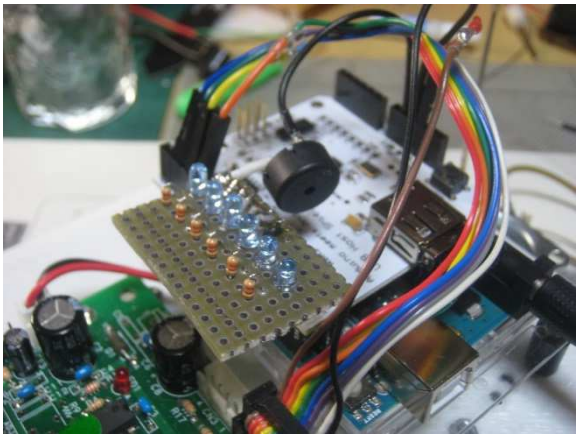
CPU: Arduino UNO

USB ホストシールド 2.0 for Arduino

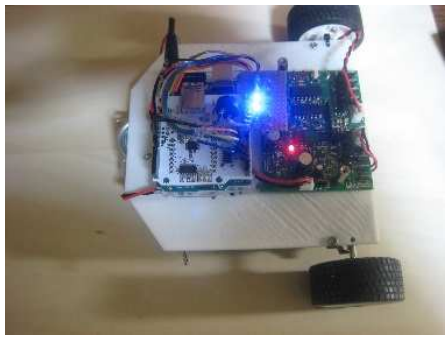
Logcool F710



モータドライブボードは mcr vol.5



MAKER UNO に合わせるため
LED とブザー追加



```
/* Simplified Logitech Extreme 3D Pro Joystick Report Parser */

/*-----------------------------------------------------------------*/
/* F710 用搬送車プログラム                                         */
/* このプログラムは Cytron HP にあった F710 を使った搬送車を */
/* Arduino に作り変えたものです。                                  */
/* モータドライブには mcr vol.5 を使用し簡易製作しました。        */
/* 速度制御(PWM)はしていません。(そのうち変更します)               */
/*                    やまがたロボットクラブ Ver 1.0 2021.11.28 */
/*-----------------------------------------------------------------*/
#include <usbhid.h>
#include <hiduniversal.h>
#include <usbhub.h>

#include "le3dp_rptparser.h"

// Satisfy the IDE, which needs to see the include statment in the ino too.
#ifdef dobogusinclude
#include <spi4teensy3.h>
#endif
#include <SPI.h>

USB Usb;
USBHub Hub(&Usb);
HIDUniversal Hid(&Usb);
JoystickEvents JoyEvents;
JoystickReportParser Joy(&JoyEvents);

#define LED2   2
#define LED3   3
#define LED4   4
#define LED5   5
#define LED6   6
#define LED7   7

#define R_Motor_disision   2
#define R_Motor_pwm        3
#define L_Motor_disision   4
#define L_Motor_pwm        5

#define BUZZER   8
```

```
#define NOTE_C4  262
#define NOTE_E4  330
#define NOTE_G4  392
#define NOTE_C5  523

int buttonXMelody[] = {NOTE_C4};
int buttonXDurations[] = {8};
int buttonAMelody[] = {NOTE_E4};
int buttonADurations[] = {8};
int buttonBMelody[] = {NOTE_G4};
int buttonBDurations[] = {8};
int buttonYMelody[] = {NOTE_C5};
int buttonYDurations[] = {8};

#define playButtonXMelody() playMelody(buttonXMelody, buttonXDurations, 1)
#define playButtonAMelody() playMelody(buttonAMelody, buttonADurations, 1)
#define playButtonBMelody() playMelody(buttonBMelody, buttonBDurations, 1)
#define playButtonYMelody() playMelody(buttonYMelody, buttonYDurations, 1)

volatile int ledPin = 0;
volatile byte F710Slider = 0x08; // Default value
volatile boolean beepFlag = false;
byte F710SliderLeft = 0x00; // Direction
byte F710SliderRight = 0x00; // Buttons
boolean F710ButtonX = false;
boolean F710ButtonA = false;
boolean F710ButtonB = false;
boolean F710ButtonY = false;

void mstop( void ) /* ロボット停止 */
{
        digitalWrite( R_Motor_disision, LOW );
        digitalWrite( R_Motor_pwm,      LOW );
        digitalWrite( L_Motor_disision, LOW );
        digitalWrite( L_Motor_pwm     , LOW );
}

void forward( void ) /* ロボット前進 */
{
        digitalWrite( R_Motor_disision, LOW  );
        digitalWrite( R_Motor_pwm,      HIGH );
        digitalWrite( L_Motor_disision, LOW  );
```

```
        digitalWrite( L_Motor_pwm,        HIGH );
}

void backword( void ) /* ロボット後進 */
{
        digitalWrite( R_Motor_disision, HIGH );
        digitalWrite( R_Motor_pwm,        HIGH );
        digitalWrite( L_Motor_disision, HIGH );
        digitalWrite( L_Motor_pwm,        HIGH );
}

void tright( void ) /* ロボット右信地旋回 */
{
        digitalWrite( R_Motor_disision, HIGH );
        digitalWrite( R_Motor_pwm,        HIGH );
        digitalWrite( L_Motor_disision, LOW  );
        digitalWrite( L_Motor_pwm,        HIGH );
}

void tleft( void ) /* ロボット左信地旋回 */
{
        digitalWrite( R_Motor_disision, LOW  );
        digitalWrite( R_Motor_pwm,        HIGH );
        digitalWrite( L_Motor_disision, HIGH );
        digitalWrite( L_Motor_pwm,        HIGH );
}

void rsla( void ) /* ロボット右ピボットターン */
{
        digitalWrite( R_Motor_disision, LOW  );
        digitalWrite( R_Motor_pwm,        LOW  );
        digitalWrite( L_Motor_disision, LOW  );
        digitalWrite( L_Motor_pwm,        HIGH );
}

void lsla( void ) /* ロボット左ピボットターン */
{
        digitalWrite( R_Motor_disision, LOW  );
        digitalWrite( R_Motor_pwm,        HIGH );
        digitalWrite( L_Motor_disision, LOW  );
        digitalWrite( L_Motor_pwm,        LOW  );
}
```

```
void rbsla( void ) /* ロボット右後ろピボットターン */
{
        digitalWrite( R_Motor_disision, LOW  );
        digitalWrite( R_Motor_pwm,      LOW  );
        digitalWrite( L_Motor_disision, HIGH );
        digitalWrite( L_Motor_pwm,      HIGH );
}

void lbsla( void ) /* ロボット左後ろピボットターン */
{
        digitalWrite( R_Motor_disision, HIGH );
        digitalWrite( R_Motor_pwm,      HIGH );
        digitalWrite( L_Motor_disision, LOW  );
        digitalWrite( L_Motor_pwm,      LOW  );
}

void setup()
{
  for (ledPin = 2; ledPin < 9; ledPin++) {
    pinMode(ledPin, OUTPUT);
  }

  Serial.begin( 115200 );
#if !defined(__MIPSEL__)
  while (!Serial); // Wait for serial port to connect - used on Leonardo, Teensy and other boards
with built-in USB CDC serial connection
#endif
  Serial.println("Start");

  if (Usb.Init() == -1)
    Serial.println("OSC did not start.");

  delay( 200 );

  if (!Hid.SetReportParser(0, &Joy))
    ErrorMessage<uint8_t>(PSTR("SetReportParser"), 1  );
}

void loop()
{
  Usb.Task();
```

```
// Direction button
    F710SliderLeft = F710Slider & 0x0F;
    if( F710SliderLeft==0x08 ) { // Released all
          mstop();
    }
    else if( F710SliderLeft==0x00 ) { // Up
       //digitalWrite(LED2, HIGH);
          forward();
    }
    else if( F710SliderLeft == 0x01 ) { // Up + Right
       //digitalWrite(LED2, HIGH);
       //digitalWrite(LED3, HIGH);
          lsla();
    }
    else if( F710SliderLeft == 0x02 ) { // Right
       //digitalWrite(LED3, HIGH);
          tleft();
    }
    else if( F710SliderLeft == 0x03 ) { // Down + Right
       //digitalWrite(LED3, HIGH);
       //digitalWrite(LED4, HIGH);
          lbsla();
    }
    else if( F710SliderLeft == 0x04 ) { // Down
       //digitalWrite(LED4, HIGH);
          backword();
    }
    else if( F710SliderLeft == 0x05 ) { // Down + Left
       //digitalWrite(LED4, HIGH);
       //digitalWrite(LED5, HIGH);
          rbsla();
    }
    else if( F710SliderLeft == 0x06 ) { // Left
       //digitalWrite(LED5, HIGH);
          tright();
    }
    else if( F710SliderLeft == 0x07 ) { // Up + Left
       //digitalWrite(LED5, HIGH);
       //digitalWrite(LED6, HIGH);
          rsla();
    }
```

```
        // A B X Y buttons
        F710SliderRight = F710Slider & 0xF0;
        if( F710SliderRight == 0x00 ) { // No buttons
            digitalWrite(LED7, LOW);
        }
        else {
            beepFlag = true;
            digitalWrite(LED7, HIGH);

            F710ButtonX = F710SliderRight & 0x10;
            F710ButtonA = F710SliderRight & 0x20;
            F710ButtonB = F710SliderRight & 0x40;
            F710ButtonY = F710SliderRight & 0x80;

            if( F710ButtonX == true ) { // Button X
                if( beepFlag == true ) {
                    beepFlag = false;
                    playButtonXMelody();
                }
            }
            if( F710ButtonA == true ) { // Button A
                if( beepFlag == true ) {
                    beepFlag = false;
                    playButtonAMelody();
                }
            }
            if( F710ButtonB == true ) { // Button B
                if( beepFlag == true ) {
                    beepFlag = false;
                    playButtonBMelody();
                }
            }
            if( F710ButtonY == true ) { // Button Y
                if( beepFlag == true ) {
                    beepFlag = false;
                    playButtonYMelody();
                }
            }
        }
}
```

```
void JoystickEvents::OnGamePadChanged(const GamePadEventData *evt)
{
  Serial.print("X: ");
  PrintHex<uint16_t>(evt->x, 0x80);
  Serial.print(" Y: ");
  PrintHex<uint16_t>(evt->y, 0x80);
  Serial.print(" Hat Switch: ");
  PrintHex<uint8_t>(evt->hat, 0x80);
  Serial.print(" Twist: ");
  PrintHex<uint8_t>(evt->twist, 0x80);
  Serial.print(" Slider: ");
  F710Slider = evt->slider;
  PrintHex<uint8_t>(F710Slider, 0x80);
  Serial.print(" Buttons A: ");
  PrintHex<uint8_t>(evt->buttons_a, 0x80);
  Serial.print(" Buttons B: ");
  PrintHex<uint8_t>(evt->buttons_b, 0x80);
  Serial.println("");

  // Clear all LED for joystick buttons
  for (ledPin = 2; ledPin < 8; ledPin++) {
    digitalWrite(ledPin, LOW);
  }
}

void playMelody(int *melody, int *noteDurations, int notesLength)
{
  for (int thisNote = 0; thisNote < notesLength; thisNote++) {
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(BUZZER, melody[thisNote], noteDuration);

    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(BUZZER);
  }
}
```