/\*

Author: Alec Korver

Date: 3/25/2017

Description:

Principia College ASME Tank Competition Arduino Mega 2560 Shield

The following code is for controlling (2) track motors and (1 or 2) tennis

ball launcher motors.

Each shield uses

(1) Custom PCB Shield

(2) Pololu VNH5019 motor drivers (5V - 24V) 12A Cont. 30A Peak 20kHz max PWM

(1) Turnigy 9x8C V2 Receiver - 8 Channel

(1) Pololu 5V S18V20F5 Voltage Regulator

To control the motors you will need (1) Turnigy 9x 8 Channel handheld

remote control. The controll will need to be synced with the reciever prior

to use. Only 3 channels are initially used for this system. There are 5

additional channels, which can be used for future expansions.

Handheld remote control info

https://hobbyking.com/media/file/725056143X2037269X20.pdf

//Good information on reading the PWM signals from the reciever

http://www.impulseadventure.com/elec/attiny-spi-rc-pwm.html

IMPORTANT NOTE ABOUT LAUNCHER

The launcher motor should be up to full speed before a tennis ball is dropped in.

This ensures that it will work similar to a pitching machine. Otherwise the launcher will be

underpowered. To increase the firing distance of the launcher there are a few options:

1)Increase the wheel inertia (increase mass of wheel)

2)Increase diameter of the launcher wheel (Larger diameter means higher speed at edge)

3)Both 1 and 2

4)Increase the voltage to the motor (Do not run continuously at higher voltage because the motor WILL burn up!!)

\*/

//CODE FOR CUSTOM CONTRLOL

#include <avr/wdt.h> //WATCHDOG TIMER

//https://tushev.org/articles/arduino/5/arduino-and-watchdog-timer

//Pins for Pololu PWM motor driver 1

//The first 4 pins need to be set either high or low

//How they are set determines whether the motor will be

//rotating CW or CCW or if it is braking or regenerating.

//See datasheet on motor driver for more details.

const int M1InA\_pin = 22;

const int M1EnA\_pin = 24;

const int M1InB\_pin = 26;

const int M1EnB\_pin = 28;

const int M1Pwm\_pin = 5;

//Pins for Pololu PWM motor driver 2

//The first 4 pins need to be set either high or low (1 or 0)

//How they are set determines whether the motor will be

//rotating CW or CCW or if it is braking or regenerating.

//See datasheet on motor driver for more details.

const int M2InA\_pin = 23;

const int M2EnA\_pin = 25;

const int M2InB\_pin = 27;

const int M2EnB\_pin = 29;

const int M2Pwm\_pin = 6;

//Pins for Turnigy 9x8C V2 Reciever

//Each pin maps to one channel of the receiver

//The input from the reciever to each pin will be a PWM signal

//The width of the PWM signal is set by the hand held remote control

//Checking this width tells us the different thumbstick and switch positions

const int Ch1\_pin = 2;

const int Ch2\_pin = 3;

const int Ch3\_pin = 18;

const int Ch4\_pin = 19;

const int Ch5\_pin = 4;

//Variables to store the PWM values read from each receiver channel(pin) - (micro seconds)

long Ch1Pwm;

long Ch2Pwm;

long Ch3Pwm;

long Ch4Pwm;

long Ch5Pwm;

//Maximum pulse width (micro seconds) - TEST AND SET THESE VALUES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

long Ch1MaxPw;

long Ch2MaxPw;

long Ch3MaxPw;

long Ch4MaxPw;

long Ch5MaxPw;

//Middle value for pulse width (micro seconds) - This is the center position for the thumbsticks - TEST AND SET THESE VALUES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

int Ch1MidPw;

int Ch2MidPw;

int Ch3MidPw;

int Ch4MidPw;

int Ch5MidPw;

//Variables for holding the PWM value (0 - 255) that we will send to the

//motor driver. This will set our left and right motor speeds

//255 is 100% output aka FULL SPEEED (FULL 24V)

int M1\_Pwm;

int M2\_Pwm;

//These are global variables which are used to transfer data from the

//ISR to the main code. These must be declared as VOLATILE.

volatile long Ch1RisingEdgeTime, Ch1FallingEdgeTime, Ch1TimeDif;

volatile long Ch2RisingEdgeTime, Ch2FallingEdgeTime, Ch2TimeDif;

volatile long Ch3RisingEdgeTime, Ch3FallingEdgeTime, Ch3TimeDif;

volatile long Ch4RisingEdgeTime, Ch4FallingEdgeTime, Ch4TimeDif;

//SETUP CODE - RUNS ONE TIME AT STARTUP TO INITIALIZE MICRO SETTINGS

void setup() {

Serial.begin(9600);//This tells our microcontroller what our clock rate is. Allows for communication protocols.

//UNCOMMENT TO USE CH1 ISR

//attachInterrupt(digitalPinToInterrupt(Ch1\_pin),Channel1DataRising,RISING);

//attachInterrupt(digitalPinToInterrupt(Ch1\_pin),Channel1DataFalling,FALLING);

//Checks Motor 1 for rising and falling edge to determine PWM length

attachInterrupt(digitalPinToInterrupt(Ch2\_pin),Channel2DataRising,RISING);

attachInterrupt(digitalPinToInterrupt(Ch2\_pin),Channel2DataFalling,FALLING);

//Checks Motor 2 for rising and falling edge to determine PWM length

attachInterrupt(digitalPinToInterrupt(Ch3\_pin),Channel3DataRising,RISING);

attachInterrupt(digitalPinToInterrupt(Ch3\_pin),Channel3DataFalling,FALLING);

//UNCOMMENT TO USE CH4 ISR

//attachInterrupt(digitalPinToInterrupt(Ch4\_pin),Channel4DataRising,RISING);

//attachInterrupt(digitalPinToInterrupt(Ch4\_pin),Channel4DataFalling,FALLING);

pinMode(M1InA\_pin,OUTPUT);

pinMode(M1EnA\_pin,OUTPUT);

pinMode(M1InB\_pin,OUTPUT);

pinMode(M1EnB\_pin,OUTPUT);

pinMode(M1Pwm\_pin,OUTPUT);

pinMode(M2InA\_pin,OUTPUT);

pinMode(M2EnA\_pin,OUTPUT);

pinMode(M2InB\_pin,OUTPUT);

pinMode(M2EnB\_pin,OUTPUT);

pinMode(M2Pwm\_pin,OUTPUT);

pinMode(Ch1\_pin,INPUT);

pinMode(Ch2\_pin,INPUT);

pinMode(Ch3\_pin,INPUT);

pinMode(Ch4\_pin,INPUT);

pinMode(Ch5\_pin,INPUT);

//Sets the motor driver to forward and reverse functions only

digitalWrite(M1EnA\_pin,HIGH);

digitalWrite(M1EnB\_pin,HIGH);

digitalWrite(M2EnA\_pin,HIGH);

digitalWrite(M2EnB\_pin,HIGH);

//Sets motor1 and motor2 to forward direction on startup

digitalWrite(M1InA\_pin,HIGH);

digitalWrite(M1InB\_pin,LOW);

digitalWrite(M2InA\_pin,HIGH);

digitalWrite(M2InB\_pin,LOW);

wdt\_enable(WDTO\_1S);

}

//MAIN LOOP - THIS WILL BE RUNNING AND LOOPING CONTINUOUSLY

void loop() {

wdt\_reset();

MotorControl();

}

//CUSTOM FUNCTIONS - ALL CODE BELOW HERE IS CUSTOM FUNCTIONS WHICH

//ARE CALLED DURING THE PROGRAM OPERATION

//https://www.arduino.cc/en/Tutorial/PWM

void MotorControl(){

//Motor 1

//If thumbstick is down

if(Ch2Pwm < Ch2MidPw){

M1\_Pwm = (255/Ch2MidPw) \* (Ch2MidPw - Ch2Pwm);

Serial.print("\n");

Serial.print("M1 Reverse: ");

Serial.print(M1\_Pwm);

Motor1Reverse();

}

//If thumbstick is in the middle

else if (Ch2Pwm == Ch2MidPw){

M1\_Pwm = 0;

analogWrite(M1Pwm\_pin,M1\_Pwm);

}

//If thumbstick is up

else if (Ch2Pwm > Ch2MidPw){

M1\_Pwm = (255/Ch2MidPw) \* (Ch2Pwm - Ch2MidPw);

Serial.print("\n");

Serial.print("M1 Forward: ");

Serial.print(M1\_Pwm);

Motor1Forward();

}

else{

}

//Motor 2

//If thumbstick is down

if(Ch3Pwm < Ch3MidPw){

M2\_Pwm = (255/Ch3MidPw) \* (Ch3MidPw - Ch3Pwm);

Serial.print("\n");

Serial.print("M2 Reverse: ");

Serial.print(M2\_Pwm);

Motor2Reverse();

}

//If thumbstick is in the middle

else if (Ch3Pwm == Ch3MidPw){

M2\_Pwm = 0;

analogWrite(M2Pwm\_pin,M2\_Pwm);

}

//If thumbstick is up

else if (Ch3Pwm > Ch3MidPw){

M2\_Pwm = (255/Ch3MidPw) \* (Ch3Pwm - Ch3MidPw);

Serial.print("\n");

Serial.print("M2 Forward: ");

Serial.print(M2\_Pwm);

Motor2Forward();

}

else{

}

//Launcher Motor Relay!!!!!!!!!!!!!!!!!!!!!!!!!!!!WRITE CODE AND ASSIGN PIN FOR RELAY CONTROL

}

//The code below is used for setting the dirction and speed of the track motors

//https://www.arduino.cc/en/Reference/digitalWrite

//https://www.arduino.cc/en/Reference/analogWrite

//Sets the direction of Motor1 to Forward

void Motor1Forward(){

digitalWrite(M1InA\_pin,HIGH);

digitalWrite(M1InB\_pin,LOW);

noInterrupts();

analogWrite(M1Pwm\_pin,M1\_Pwm);

interrupts();

}

//Sets the direction of Motor2 to Forward

void Motor2Forward(){

digitalWrite(M2InA\_pin,HIGH);

digitalWrite(M2InB\_pin,LOW);

noInterrupts();

analogWrite(M2Pwm\_pin,M2\_Pwm);

interrupts();

}

//Sets the direction of Motor1 to Reverse

void Motor1Reverse(){

digitalWrite(M1InA\_pin,LOW);

digitalWrite(M1InB\_pin,HIGH);

noInterrupts();

analogWrite(M1Pwm\_pin,M1\_Pwm);

interrupts();

}

//Sets the direction of Motor2 to Reverse

void Motor2Reverse(){

digitalWrite(M2InA\_pin,LOW);

digitalWrite(M2InB\_pin,HIGH);

noInterrupts();

analogWrite(M2Pwm\_pin,M2\_Pwm);

interrupts();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*INTERRUPT SERVICE ROUTINE FUNCTIONS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//External ISRs are ideal to use because they are called whenever an external event happens

//This makes it unlikely for us to miss our signal. Otherwise we would have to constantly poll and check

//The input pins for signal changes.

//http://www.engblaze.com/we-interrupt-this-program-to-bring-you-a-tutorial-on-arduino-interrupts/

//ISR Function to read the PWM signal from Channel 1

void Channel1DataRising(){

Ch1RisingEdgeTime = micros();

}

void Channel1DataFalling(){

Ch1FallingEdgeTime = micros();

Ch1TimeDif = Ch1FallingEdgeTime - Ch1RisingEdgeTime;

Ch1Pwm = Ch1TimeDif;

}

//ISR Function to read the PWM signal from Channel 2

void Channel2DataRising(){

Ch2RisingEdgeTime = micros();

}

void Channel2DataFalling(){

Ch2FallingEdgeTime = micros();

Ch2TimeDif = Ch2FallingEdgeTime - Ch2RisingEdgeTime;

Ch2Pwm = Ch2TimeDif;

}

//ISR Function to read the PWM signal from Channel 3

void Channel3DataRising(){

Ch3RisingEdgeTime = micros();

}

void Channel3DataFalling(){

Ch3FallingEdgeTime = micros();

Ch3TimeDif = Ch3FallingEdgeTime - Ch3RisingEdgeTime;

Ch3Pwm = Ch3TimeDif;

}

//ISR Function to read the PWM signal from Channel 4

void Channel4DataRising(){

Ch4RisingEdgeTime = micros();

}

void Channel4DataFalling(){

Ch4FallingEdgeTime = micros();

Ch4TimeDif = Ch4FallingEdgeTime - Ch4RisingEdgeTime;

Ch4Pwm = Ch4TimeDif;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END OF INTERRUPT SERVICE ROUTINE FUNCTIONS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*