/\* mbed Microcontroller Library

\* Copyright (c) 2019 ARM Limited

\* SPDX-License-Identifier: Apache-2.0

\*/

#include "mbed.h"

#include "platform/mbed\_thread.h"

#include <string.h>

#include <PwmOut.h>

#include <SPISlave.h>

/\*Liaison SPI avec RPI\*/

//SPISlave device(PB\_5, PB\_4, PB\_3, PA\_4); //mosi, miso, sclk, ssel pour F411RE

SPISlave device(PB\_5, PB\_4, PB\_3, PA\_4); //mosi, miso, sclk, ssel pour L432KC

/\*PWM Moteur et Servo\*/

PwmOut motor\_pwm(D9);

PwmOut steer\_pwm(D6);

Serial serial\_pc(USBTX, USBRX);

/\*Liaison capteur optique\*/

//InterruptIn capteur\_axe(PA\_0); //pour F411RE

InterruptIn capteur\_axe(PA\_4); //pour L432KC

Timer t;

Ticker flipper;

InterruptIn capteur\_proxG(PA\_0);

InterruptIn capteur\_proxD(PA\_3);

uint8\_t back\_capt = 0;

uint32\_t back\_capt\_time;

uint32\_t t\_now;

uint32\_t t\_old;

uint32\_t dt;

uint16\_t vit\_mes\_now;

uint16\_t vit\_mes\_old;

/\*FONCTIONS\*/

void fct\_capteur\_axe(void)

{

t\_now = t.read\_us();

dt = t\_now - t\_old;

vit\_mes\_now = 8235294/dt;//Convert into cm/s

t\_old = t\_now;

}

void fct\_capteur\_prox(void)

{

back\_capt = 1;

back\_capt\_time = t.read();

}

void check(void)

{

static uint32\_t check\_dt\_now = 0;

if (dt == check\_dt\_now){

vit\_mes\_now = 0;

}

check\_dt\_now = dt;

vit\_mes\_old = vit\_mes\_now;

}

uint32\_t steer\_angle\_2\_pulse\_width( int32\_t steer\_angle );

uint32\_t motor\_speed\_2\_pulse\_width( int32\_t motor\_speed );

int main()

{

//-------------------------- PWM Control Set-up --------------------------//

char carac\_recu;

char text[40];

uint32\_t steer\_pulse\_width = 2300;

uint32\_t motor\_pulse\_width = 1500;

int32\_t steer\_angle = 0; //en ° allant de -15° à +15°

int32\_t motor\_speed = 0; //allant de 0 à 10

uint32\_t recep, recep1;

/\*Pour le servo HS 645MG :

- angle = -15° max right <-- steer\_pulse\_width = 2560

- angle = 0° <-- steer\_pulse\_width = 2300

- angle = 15° max left <-- steer\_pulse\_width = 2050 \*/

//--------------------------- Serial Set-up ------------------------------//

serial\_pc.baud(115200);

serial\_pc.printf("TER - VROOM VROOM : Initialisation \n\r");

//----------------------------- SPI Set-up -------------------------------//

device.frequency(1000000);

device.format(32,0);

motor\_pwm.pulsewidth\_us(motor\_pulse\_width);

steer\_pwm.pulsewidth\_us(steer\_pulse\_width);

//----------------- Speed measurement and check Set-up -------------------//

t.start();

capteur\_axe.fall(&fct\_capteur\_axe);

capteur\_proxG.rise(&fct\_capteur\_prox);

capteur\_proxD.rise(&fct\_capteur\_prox);

flipper.attach(&check, 0.1);

while (true) {

if (t.read() > (back\_capt\_time +1))

{

back\_capt = 0;

}

if(device.receive())

{

recep = device.read();

device.reply(vit\_mes\_now);

recep1 = device.read();

device.reply(back\_capt);

if(recep < 117)

{

motor\_speed = recep;

}

else if (recep > 117)

{

steer\_angle = recep - 140;

}

if(recep1 < 117)

{

motor\_speed = recep1;

}

else if (recep1 > 117)

{

steer\_angle = recep1 - 140;

}

printf("recep = %d motor\_speed = %d recep1 = %d steering = %d \n", recep, motor\_speed, recep1, steer\_angle);

//serial\_pc.printf("angle = %d speed = %d motor pulsewidth = %d\n", steer\_angle, motor\_speed, motor\_pulse\_width);

steer\_pulse\_width = steer\_angle\_2\_pulse\_width(steer\_angle);

steer\_pwm.pulsewidth\_us(steer\_pulse\_width);

motor\_pulse\_width = motor\_speed\_2\_pulse\_width(motor\_speed);

motor\_pwm.pulsewidth\_us(motor\_pulse\_width);

printf("motor\_pulse\_width = %d vit\_mes\_nom = %d back\_capt = %d \n\r", motor\_pulse\_width, vit\_mes\_now, back\_capt);

}

}

}

uint32\_t steer\_angle\_2\_pulse\_width( int32\_t steer\_angle )

{

uint32\_t steer\_pulse\_width;

steer\_pulse\_width = (uint32\_t)((2570-2030)/40 \* steer\_angle + 2300);

return steer\_pulse\_width;

}

uint32\_t motor\_speed\_2\_pulse\_width( int32\_t motor\_speed )

{

uint32\_t motor\_pulse\_width;

if (motor\_speed >= 10) //FORWARD

{

if (motor\_speed == 115)

{

motor\_pulse\_width = 1590;

}

else

{

motor\_pulse\_width = (uint32\_t)((1440-1480) \* (motor\_speed-10)/100 + 1480);

//max forward speed pulsewidth = 1200

}

}

else if (motor\_speed < 10) //BACKWARD

{

motor\_pulse\_width = (uint32\_t)((1700-1590)/10 \* motor\_speed + 1590);

//max backward speed pulsewidth = 1900

}

return motor\_pulse\_width;

}