Arduino C og Robot C kode

Arduino C kode

```
//******************LegoCarduino**********************


int var[] = {3,4,5,6,7,8,9,10};            //bestemmer hvilken plass i arrayet portene skal stå på

//navngir variabler og setter variablene til sine ulike plasser og porter i arrayet
int hovedRod = var[0], hovedGul = var[1], hovedGronn = var[2], sideRod = var[3], sideGul =
var[4];
int sideGronn = var[5], goendeRod = var[7], goendeGronn = var[6];
int hovedKnappVerdi = 2;
int sensTrig = 12;
int sensEcho = 11;
int i = 0;                                 //setter telleren "i" lik 0


//setter de ulike fargene til fargenavn som ikke kan endre verdi gjennom programmet

#define gronn 0
#define gul 1
#define rod 2
#define gulrod 3
#define gronnBlink 4



void setup()
{
        Serial.begin(9600);
         for(i=0;i<8;i++)
        {
                pinMode(var[i], OUTPUT);
                 //setter pinne 3,4,6,7,8,9,10,11 til å være utganger
         }

        pinMode(hovedKnappVerdi, INPUT_PULLUP);
        //setter pinne 2 til å være en inngang med pullup slik at knappen ikke blir følsom for
små forstyrrelser
        pinMode(sensEcho, INPUT);
        //setter den ene pinnen til sensoren som inngang(leser signaler som sendes ut av
sensTrig)
        pinMode(sensTrig, OUTPUT);
         //setter den andre pinnen til sensoren som utganger(sender ut signaler som leses av
sensEcho)
}
```

```
//funksjonen skrur av alle lys for hovedgate, for deretter å sette lyset grønt, rødt , gulrødt eller gult, ettersom hvilken farge som angis når funksjonen kalles fra void loop

void settHovedLys(int farge)
{
	digitalWrite(hovedGronn, LOW);
	digitalWrite(hovedRod, LOW);
	digitalWrite(hovedGul, LOW);
	if (farge == gronn)
	{
		digitalWrite(hovedGronn, HIGH);
	}
	else if(farge == gul)
	{
		digitalWrite(hovedGul, HIGH);
	}
	else if (farge == rod)
	{
		digitalWrite(hovedRod, HIGH);
	}
	else if (farge == gulrod)
	{
		digitalWrite(hovedGul, HIGH);
		digitalWrite(hovedRod, HIGH);
	}
}

//funksjonen skrur av alle lys for sidegate, for deretter å sette lyset grønt, rødt, gulrødt eller gult, ettersom hvilken farge som angis når funksjonen kalles fra void loop

void settSideLys(int farge)
{
	digitalWrite(sideGronn, LOW);
	digitalWrite(sideGul, LOW);
	digitalWrite(sideRod, LOW);
	if (farge == gronn)
	{
		digitalWrite(sideGronn, HIGH);
	}
	else if (farge == gul)
	{
		digitalWrite(sideGul, HIGH);
	}
	else if (farge == rod)
```

```
        {
        digitalWrite(sideRod, HIGH);
        }
        else if (farge == gulrod)
        {
        digitalWrite(sideRod, HIGH);
        digitalWrite(sideGul, HIGH);
        }
}

//funksjonen skrur av begge lys for gående, for deretter å sette lyset grønt, rødt eller blinke grønt, ettersom hvilken farge som angis når funksjonen kalles fra void loop

void settGoendeLys(int farge)
{
        digitalWrite(goendeGronn, LOW);
        digitalWrite(goendeRod, LOW);
        if (farge == gronn)
        {
                digitalWrite(goendeGronn, HIGH);
        }
        else if (farge == rod)
        {
                digitalWrite(goendeRod, HIGH);
        }
        else if (farge == gronnBlink)
        {
                int teller = 0;
                while (teller < 8)
                {
                        digitalWrite(goendeGronn,HIGH);
                        delay(200);
                        digitalWrite(goendeGronn,LOW);
                        delay(200);
                        teller = teller+1;
                }
        }
}


//en funskjon som tar inn tre verdier som representerer, rødt, gult eller grønt lys i hoved-lyskrysset og sender verdiene videre til nxt-hjernen over bluetooth

void bluetooth(int lys)
{
        if(lys==1)        //hvis lyset er grønt som tilsvarer verdien 1, gå inn i if-setningen
```

```
        {
                byte melding1[] = {10, 0, 128, 9, 0, 6, 1, 0, 0, 0, 0, 0};
        //setter variabelen melding1 til datatypen byte og tilegner variabelen melding1 et array
        med 12 bytes med 12 plasser fra 0-11
                Serial.write(melding1,12);
         //sender arraeyet med 12 bytes som representerer tallet 1 over bluetooth til nxt-hjernen
                delay(200);      //venter 200 milisekunder
        }

        if (lys==2)       //hvis lyset er rødt som tilsvarer verdien 2, gå inn i if-setningen
        {
                byte melding2[] = {10, 0, 128, 9, 0, 6, 2, 0, 0, 0, 0, 0};
                int  teller = 0;

                while(teller < 2)
                {
                        Serial.write(melding2,12);
                        //sender arraeyet med 12 bytes som representerer tallet 2 over bluetooth
                til nxt-hjernen
                         delay(100);    //venter 200 milisekunder
                        Serial.write(melding2,12);
                        delay(100);
                        teller = teller + 1;
                }
        }

         if (lys==3)      //hvis lyset er gult som tilsvarer verdien 3, gå inn i if-setningen
        {
                byte melding3[] = {10, 0, 128, 9, 0, 6, 3, 0, 0, 0, 0, 0};
                int  teller = 0;

                while(teller < 2)
                {
                        Serial.write(melding3,12);
                        //sender arraeyet med 12 bytes som representerer tallet 2 over bluetooth
                til nxt-hjernen
                         delay(100);    //venter 200 milisekunder
                        Serial.write(melding3,12);
                        delay(100);
                        teller = teller + 1;
                }
        }
}


void loop()
```

```
{
        long avstand;
        long varighet;
        int standardTilstand = 0;
        int t = 0;

/*
while-løkka gjør at programmet settes i standardtilstand med hovedlys grønt, side-og gåendelys rødt
sjekker hele tiden avstand til sensoren eller om knappen trykkes på
*/
        while(standardTilstand == 0)
        {
                settHovedLys(gronn);
                bluetooth(1);
                settGoendeLys(rod);
                settSideLys(rod);
                int knappVerdi = digitalRead(2);

//måler og regner ut avstand til eventuelle objekter med avstandssensoren

                digitalWrite(sensTrig, LOW);
                delay(2);
                digitalWrite(sensTrig, HIGH);
                delay(10);
                digitalWrite(sensTrig, LOW);
                varighet = pulseIn(sensEcho, HIGH);
                avstand = (varighet/2)/29.1;            //avstand måles i cm

 //når knappen trykkes eller et objekt har en avstand mindre enn ca.5cm fra sensoren, så starter programmet
                if (avstand < 5)
                {
                        standardTilstand = 1;
                }
                if(knappVerdi == LOW)
                {
                        standardTilstand= 1;
                }
        }

/*kaller de ulike funksjonene som endrer farge på lyskryssene og sender bluetooth-signaler.
Denne sekvensen bytter hovedlyset til rødt, og sidelys samt gående lys til grønt.
Når hovedlyset endrer seg fra grønt til gult og fra gult til rødt, sendes bluetooth signalene, 1,2 og 3 til NXT-roboten.
Tallene 1-3 tilsvarer: Grønt lys(1), gult lys(2) og rødt lys(3).
```

```
På slutten av sekvensen går lyskrysset tilbake til standardtilstand: Hovedlys grønt, sidelys
samt gående lys rødt.
*/

        delay(500);
        settHovedLys(gul);
        for(t=0;t<7;t++)
        {
                bluetooth(3);
        }
        settHovedLys(rod);
        bluetooth(2);
        settSideLys(gulrod);
        for(t=0;t<6;t++)
        {
                bluetooth(2);
        }
        t=0;
        settSideLys(gronn);
        settGoendeLys(gronn);
        for(t=0;t<16;t++)
        {
                bluetooth(2);
        }
        t=0;
        settSideLys(gul);
        settGoendeLys(gronnBlink);
        settSideLys(rod);
        settGoendeLys(rod);
        settHovedLys(gulrod);
        for(t=0;t<6;t++)
        {
                bluetooth(3);
        }

        knappVerdi = LOW;
}

/*
når sekvensen er kjørt, går knappen/sensoren tilbake til standardtilstand,
  og sekvensen vil ikke kjøre før sensoren eller knappen trykkes igjen
*/
```

Robot C kode

```
#pragma config(Sensor, S1,    sensCenter,    sensorCOLORFULL)
#pragma config(Sensor, S2,    sensLeft,    sensorCOLORFULL)
#pragma config(Sensor, S3,    distFront,    sensorSONAR)
#pragma config(Sensor, S4,    sensRight,    sensorCOLORFULL)
#pragma config(Motor,  motorA,          dir,           tmotorNXT,     PIDControl,    encoder)
#pragma config(Motor,  motorB,          right,         tmotorNXT,     PIDControl,    encoder)
#pragma config(Motor,  motorC,          left,          tmotorNXT,     PIDControl,    encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
//sensCenter, sensorCOLORFULL kan erstattes med distSide, sensorSONAR ved oenske
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//
//                                  LegoCarduino
//
//                                  Bilens kode
//
// motor[motorA] = motor[dir] justerer retning
// motor[motorB] = motor[right] justerer fart paa begge bakhjulene
// !!! NB! Husk aa koble bilen til Bluetooth Adafruit EZ-Link !!!
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//definerer oeverst variablene som brukes i hele programmet
int speedTurn = 80;             //farten til motoren som justerer retningen
int rotation;                   //maaling av hvor mange grader motoren som justerer
int degTurn = 360/5;            //grader for motor[dir]
int blckLineColour = 1;         // = 1 = svart i forhold til RGB sensorene
int redLineColour = 5;          // = 5 = roedt i forhold til sensorene
Int colourCenter;               //hva sensCenter ser, i tilfelle man bruker tre RGB sensorer
int colourRight;                //hva sensRight ser
int colourLeft;                 //hva sensLeft ser
int speedMtr = 50;              //standard fart paa bak motorene
int obstacle;                   //avstand sensor foran
//int around;                   //avstand sensor paa side, i tilfelle man bruker to
avstandssensorer
int bluetoothmelding = 1;
bool direction[3] = {false, true, false};               //bestemmer hvor bilen skal, rett fram
som default
bool noProb = true;             //noProb er true saa langt det IKKE er en kjegle paa veien er det
en kjegle er noProb false
bool traffic = false;           //traffic sjekker bluetooth meldinger

//Reset alt til nullverdier for sikkerhets skyld
void resetAll()
{
```

```
	nMotorEncoder[dir] = 0;		//setter rotation lik 0, for sikkerhetsskyld
	nSyncedMotors = synchNone;		//no synch (OBLIGATORISK foer synchBC), i tilfelle man oensker aa synkronisere en motor til en annen
	nSyncedMotors = synchBC;	//synkroniserer motor C til motor B: motor C blir paa en maate slave av motor B, og skal gjoere alt motor B gjoer
}

//Denne funksjonen leser verdiene, og avgjoer om det er f.eks. hindringer
//funksjonen skriver ogsaa ut paa kjermen viktige info
task readVar()
{
	while (true)
	{
		//Variablene som oppdateres hele tiden...
		rotation = nMotorEncoder[dir];
		//colourCenter = SensorValue[sensCenter]; //i tilfelle man bruker tre RGB sensorer
		colourRight = SensorValue[sensRight];
		colourLeft = SensorValue[sensLeft];
		obstacle = SensorValue[distFront];
		//around = SensorValue[distSide]; //i tilfelle man bruker to avstandssensorer
		bluetoothmelding = message;

		//...blir skrivet ut
		nxtDisplayCenteredTextLine(0, "dist Front: %d", obstacle);
		//nxtDisplayCenteredTextLine(1, "dist Side: %d", around);  //i tilfelle man bruker to avstandssensorer
		nxtDisplayCenteredTextLine(2, "Lft:%d - Rght:%d", colourLeft, colourRight);
		//nxtDisplayCenteredTextLine(3, "Center: %d", colourCenter);	//i tilfelle man bruker tre RGB sensorer
		//Retningen
		if (colourLeft == blckLineColour)
		{
			nxtDisplayCenteredTextLine(4, "direction: Left");
			direction[0] = true;
			direction[1] = false;
			direction[2] = false;
		}
		if (direction[1] == true)
		{
			nxtDisplayCenteredTextLine(4, "direction: Cntr");
		}
		if (colourRight == blckLineColour)
		{
			nxtDisplayCenteredTextLine(4, "direction: Right");
			direction[0] = false;
```

```
                direction[1] = false;
                direction[2] = true;
            }

            //Bluetooth traffic lights
            if (bluetoothmelding == 1 && traffic == true)          //Groent lys
            {
                nxtDisplayCenteredTextLine(3, "Trfc lght: Green");
            }
            if (bluetoothmelding == 2 && traffic == true)          //Roedt lys
            {
                nxtDisplayCenteredTextLine(3, "Trfc lght: Red");
            }
            if (bluetoothmelding == 3 && traffic == true)          //Gult lys
            {
                nxtDisplayCenteredTextLine(3, "Trfc lght: Yellow");
            }
            if (colourLeft == redLineColour || colourRight == redLineColour)
            {
                traffic = true;
            }
            if (colourLeft != redLineColour && colourRight != redLineColour)
            {
                nxtDisplayCenteredTextLine(6, "not reading BT");
                traffic = false;
            }

            //Avstandssmaaling
            if (obstacle > 20)
            {
                noProb = true;
                nxtDisplayCenteredTextLine(7, "no obstacle");
            }
            if (obstacle <= 20)
            {
                noProb = false;
                nxtDisplayCenteredTextLine(7, "obstacle detected!");
            }
        }
}

task turnRight ()
{
        while (true)
        {
            while (direction[2] == true && rotation < 0.9*degTurn)
```

```
		{
			motor[dir] = speedTurn;
		}
		while  (direction[2] == true && rotation >= 0.9*degTurn)
		{
			motor[dir] = 0;
		}
	}
}

task turnLeft ()
{
	while (true)
	{
		while (direction[0] == true && rotation > 0.9*(-degTurn))
		{
			motor[dir] = -speedTurn;
		}
		while (direction[0] == true && rotation <= 0.9*(-degTurn))
		{
			motor[dir] = 0;
		}
	}
}

task steerUp ()
{
	while (true)
	{
			while (direction[1] == true && rotation > 10)
			{
				motor[dir] = -speedTurn;
			}
			while (direction[1] == true && rotation < -10)
			{
				motor[dir] = speedTurn;
			}
			while (direction[1] == true && rotation <= 10 && rotation >= -10)
			{
				motor[dir] = 0;
			}
	}
}

//BT aktiveres kun nr bilen leser roedt stripe, ellers kjører bilen som normalt
task drive() //1 = green, 2 = red, 3 = yellow;
```

```
{
 while(true)
 {
        while (noProb == false)
        {
                motor[right] = 0;
        }
        while (traffic == false && noProb == true)
        {
                ClearMessage();
                motor[right] = speedMtr;
        }
        while (traffic == true && noProb == true)              // oppdager roed stripe = er i
naerheten av lysskrysset
        {
                if(bluetoothmelding == 1) //Green light
                {
                        motor[right] = speedMtr;
                        wait1Msec(100);
                        ClearMessage();
                }

                if(bluetoothmelding == 2)      //Red light
                {
                        motor[right] = 0;
                        wait1Msec(100);
                        ClearMessage();
                }

                if(bluetoothmelding == 3)      //Yellow light
                {
                        motor[right] = speedMtr/2;
                        wait1Msec(100);
                        ClearMessage();
                }
        }
 }
}

task main()
{
        nxtDisplayCenteredBigTextLine(2, "HELLO");
        wait1Msec(1000);
        nxtDisplayCenteredBigTextLine(4, "STARTING");
        wait1Msec(1000);
        nxtDisplayClearTextLine(1);
```

```
	nxtDisplayClearTextLine(2);
	nxtDisplayClearTextLine(3);
	nxtDisplayClearTextLine(4);
	resetAll();
	StartTask (readVar);
	StartTask (drive);
	StartTask (steerUp);
	StartTask (turnRight);
	StartTask (turnLeft);
	while (true)
	{
		wait1Msec(100);
	}
}
```