Mașină cu Joystick Controller

Introducere

Nu am bani de mașina visurilor mele, așa că m-am hotărât să o fac cu mâna mea.

O idee simplă care va da startul unei munci ce va avea drept rod o nouă jucărie pentru sora mea mai mică. Mașinuța electrică pe care vreau să o proiectez nu doar că este eco, dar e făcută și din plastic nereciclat. Acționată de joysticks și imitând condusul din viața reală, este visul pe care îl pot trăi la scară mică.

Descriere generală

Visul meu la scară mică este alcătuit din două părți:

- 1. Mașină
- 2. Controller cu Joystick

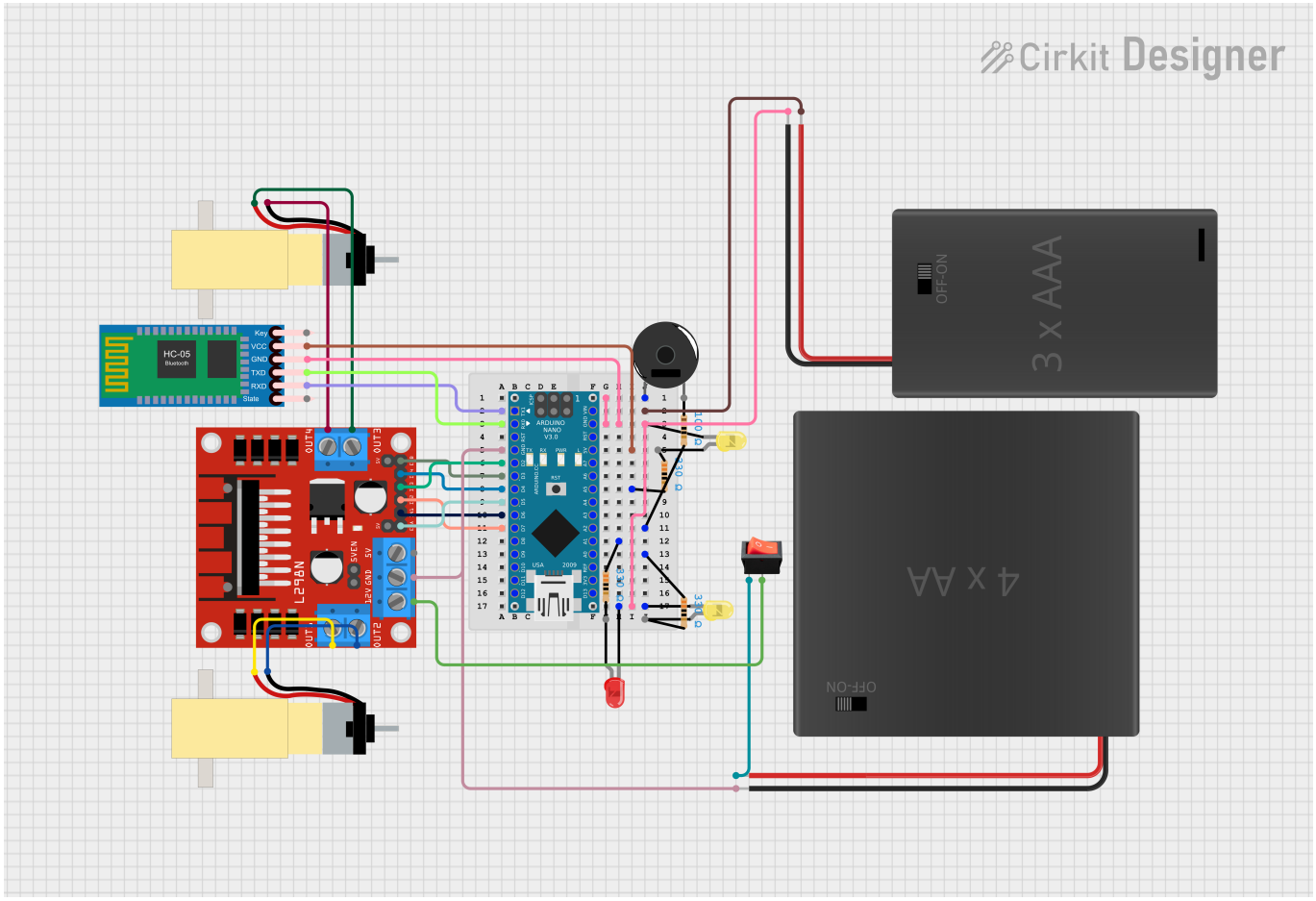
Cele două comunică prin module Bluetooth, unidirecțional. Controller-ul Joystick are funcție de input și transmite comenzile de la jucător mașinii, parte care servește de output. Comunicarea inversă nu este necesară.

Direcția și sensul de mers sunt date de un joystick, restul funcționalităților fiind accesibile printr-o serie de butoane alăturate. Aceste funcționalități includ semnalizarea luminoasă și sonoră, folosite mai mult sau mai puțin de majoritatea șoferilor bucureșteni.

Hardware Design

Partea de controller remote presupune un Arduino Uno, shielduit cu modulul joystick și butoanele aferente Funduino. Acest shield se va alimenta la sursa de 6V, obținută prin atașarea a 4 baterii AA într-un suport. Fiind un input de peste 5V, Uno va face output pe pinul de 5V, ce va fi folosit ca să alimenteze modulul Bluetooth cu rol master. Cum nu am reușit să descopăr modul de utilizare a pinilor sub eticheta de 'Bluetooth' (sau mi-am luat țeapă și chiar nu merg), am folosit pinii originali de RX/TX pentru comunicarea serială cu mașina.

În ce privește mașina propriu zisă, ea are ca nucleu un Arduino Nano, care operează strict ca output. El primește date serial de la modulul Bluetooth atașat, care este configurat drept slave în comunicare. Alimentarea este realizată cu 7 baterii AA, separate în doi suporți, pentru a asigura performanța de durată a mașinii: 4 baterii AA (6V) pentru cele două motoare cu reductor (3-6V), respectiv 3 baterii AA (4.5V) pentru alimentarea plăcuței Nano (3.3V minim), care la rândul ei alimentează modulul HC-05 (3.3V - 5V). Tot de plăcuța Nano sunt atașate 3 diode LED pentru semnalizarea luminoasă și un buzzer cu rol de claxon.



Bluetooth Pairing

Pentru a face pair la cele două module HC-05, am trecut printr-o serie de etape și încercări la diferite lungimi de bandă. În documentația originală, se impune default o funcționare pe 9600 baud, configurarea realizându-se la 38400 baud. Din nefericire, 9600 de biți pe secundă s-au dovedit a fi prea puțin pentru un transfer smooth de date, așa că am ridicat cota până la 38400 de biți pe secundă. Diferențele observate au fost evidente, dar nu am îndrăznit să merg mai departe la o rată mai mare, deoarece unul dintre modulele Bluetooth pe care le-am achiziționat s-a dovedit a fi mai vechi și mi-am cam luat țeapă, dar 38400 baud este chiar decent ca viteză.

Procesul în sine a constat în trecerea a modului de configurare în cazul ambelor module, master pentru controller și slave pentru mașină. Intrând în slave, am copiat adresa MAC a device-ului și am schimat rata de transfer la 38400 baud, față de valoarea default de 9600 baud. Comenzile folosite au fost următoarele:

```
AT+NAME=BTSLAVE
```

```
AT+ADDR?
AT+UART=38400,0,0
```

Pentru master, am trecut prin mai multe configurări, după cum se vede mai jos:

```
AT+NAME=BTMASTER
AT+ROLE=1
AT+CMODE=0
AT+BIND=adresa,mac,device
AT+UART=38400,0,0
```

În urma configurării software (realizată ba în monitorul serial oferit de Arduino IDE, ba cu utilitarul CoolTerm), am deconectat modulele ca acestea să-și dea un restart, ca mai apoi să intre în modul de discovery și master să îl găsească pe slave. Cele două module Bluetooth clipesc sincron și sunt gata să fie folosite.

Pentru a adăuga o notă de realitate proiectului și pentru a-mi diferenția mașina de un BMW trivial, am adăugat leduri cu rol de semnalizare luminoasă stânga-dreapta și un led dedicat mersului marșalier, care se activează automat când mașina dă cu spatele. Cele 3 leduri sunt conectate la pini analog, pentru a produce un clipit incremental ce nu deranjează la ochi. Mai multe sunt explicate la rubrica următoare.

Software Design

Ca mediu de dezvoltare am folosit Arduino IDE exclusiv. Nu am folosit librării sau surse third-party, deoarece codul în sine este destul de simplu pentru a facilita comunicarea între cele două plăcuțe Arduino prin modulele HC-05.

În primul rând, mi-am declarat pinii de lucru. În cazul unității master, am avut exclusiv pini de INPUT, unitatea slave servind de OUTPUT.

```
// MASTER CODE
// -----------
#define PIN_ANALOG_X 0
#define PIN_ANALOG_Y 1

int A = 2;
int B = 3;
int C = 4;
int D = 5;
int E = 6;
int F = 7;

void setup() {
  pinMode(A, INPUT);
  pinMode(B, INPUT);
```

```
  pinMode(C, INPUT);
  pinMode(D, INPUT);
  pinMode(E, INPUT);
  pinMode(F, INPUT);

  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);

  Serial.begin(38400);

  delay(100);
}
```

```
// SLAVE CODE
// ----------
// MOTOR PINS
int ena = PD5;
int in1 = PD7;
int in2 = PD6;
int in3 = PD4;
int in4 = PD2;
int enb = PD3;

int ledLeft = A5;
int ledRight = A0;
int ledBack = A1;

int buzzer = A2;

void setup() {
  pinMode(ena, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(enb, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  pinMode(ledLeft, OUTPUT);
  pinMode(ledRight, OUTPUT);
  pinMode(ledBack, OUTPUT);

  analogWrite(ledLeft, 0);
  analogWrite(ledRight, 0);
  analogWrite(ledBack, 0);

  pinMode(buzzer, OUTPUT);
```

```
  noTone(buzzer);

  Serial.begin(38400);
  delay(100);
}
```

După cum se observă mai sus, comunicarea serială am început-o pe 38400 baud, fiind cea mai bună opțiune a mea. Monitorul serial default al Arduino IDE mi-a fost suficient, așa că nu am importat librăria SoftwareSerial pentru a-mi crea propriul monitor inutil. Modul în care funcționează această comunicare este prin trimiterea de charactere unice într-un moment de timp. Acest caracter reprezintă o stare, ce este ulterior procesată de slave și pe baza ei se realizează sau nu ceva.

M-am folosti de acest exemplu de pe forumul oficial Arduino pentru primirea caracterelor. Atunci când se receptează un influx de date, variabila newData ia valoarea true și se intră în codul de verificare a stării primite de la master.

```
if (Serial.available() > 0) {
  state = Serial.read();
  newData = true;
}

if (newData == true) {
  Serial.print("This just in ... ");
  Serial.println(state);
  newData = false;
}
```

Considerând faptul că trimiterea de numere întregi ca valori este suficient de complicată și nu am simțit nevoia să o implementez, am ales să mapez joystick-ul în 8 zone majore de referință, pe baza cărora se va deplasa mașina, așa cum se observă în codul de mai jos:

```
if (Y > 530 && (X > 400 && X < 600)) { // move straight forward
  Serial.print(1);
} else if (Y < 500 && (X > 400 && X < 600)) { // move straight backward
  Serial.print(2);
} else if ((Y > 400 && Y < 600) && X > 530){ // rotate narrow right forward
  Serial.print(3);
} else if ( . . . ) {
  . . .
} else {
  Serial.print(0);
}
```

Starea 0 (48 în ASCII) este cea default, în care mașina nu se mișcă deloc și nu are loc nicio schimbare de la starea anterioară. Mai jos las o listă de stări valide în cod, care sunt responsabile de partajarea unei instrucțiuni:

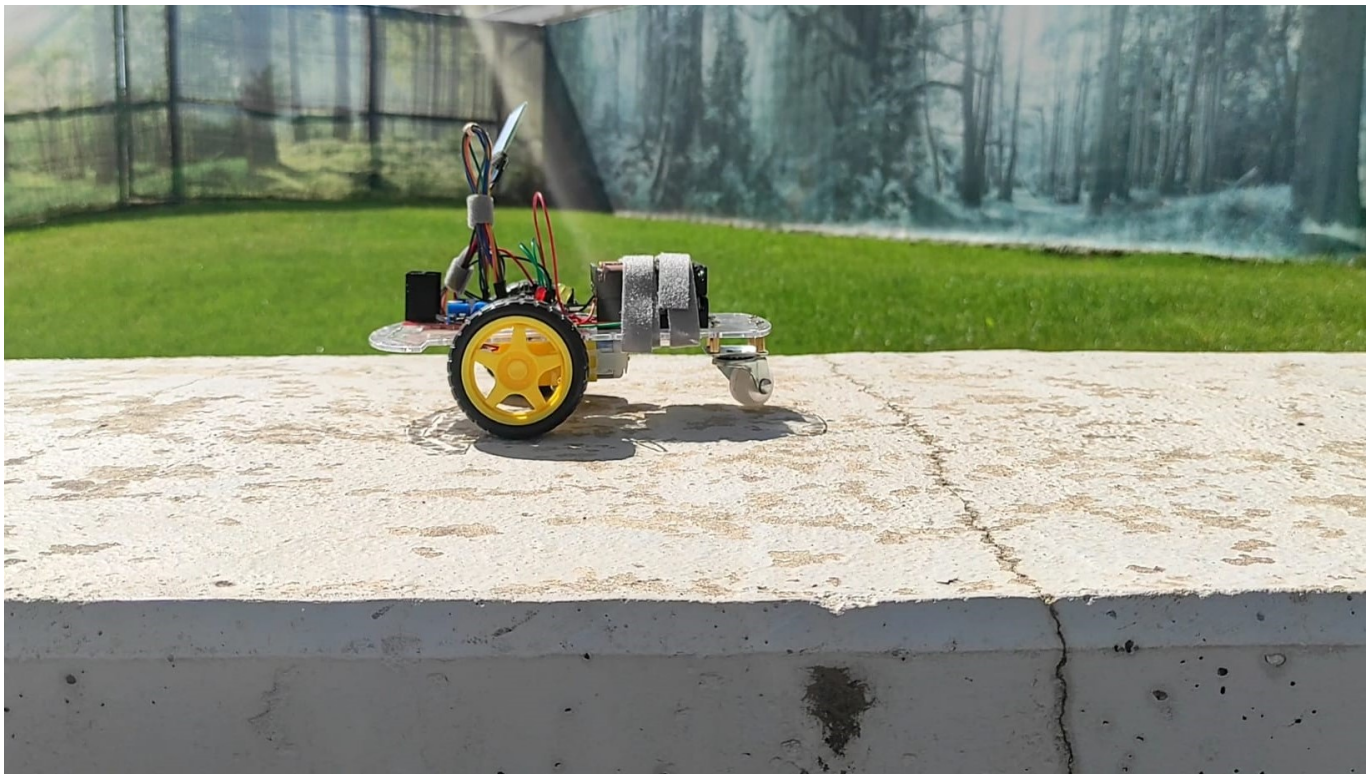
- 0, stare de repaus
- 1 - 2, stări de mișcare înainte - înapoi
- 3 - 4, stări de rotire dreapta - stânga pe loc, direcția înainte
- 5 - 6, stări de rotire dreapta - stânga larg, direcția înainte

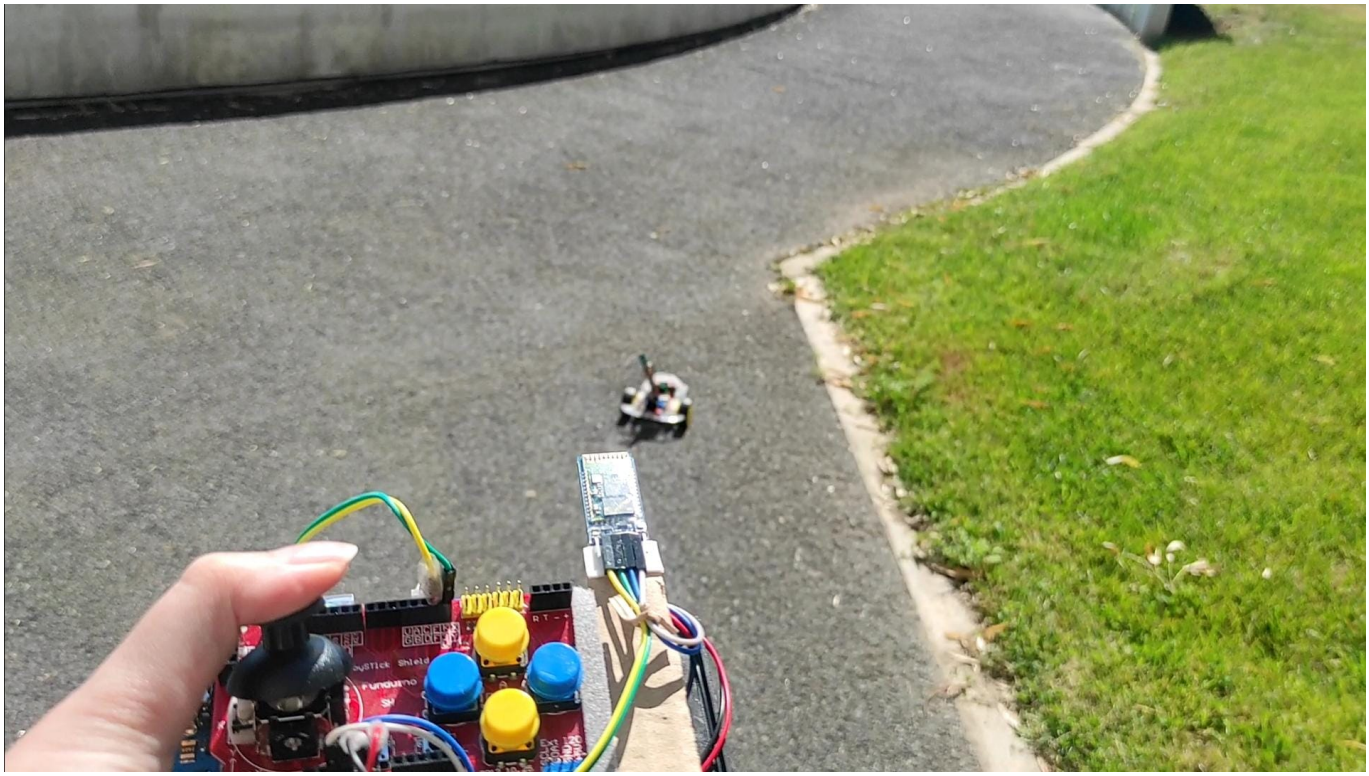
- 7 - 8, stări de rotire dreapta - stânga larg, direcția înapoi
- a - f, stări date de butoane, în ordine:
  - A, claxon
  - B, semnalizare dreapta
  - C, -
  - D, semnalizare stânga
  - E, oprire semnalizare
  - F, oprire claxon

Deoarece am ales comunicare printr-un singur caracter trimis o dată, este nevoie de un terminator al comenzilor prin butoane. Din acest motiv, am ales să folosesc butoanele E și F din shield. Alternativ, aș fi putut implementa un reader mai deștept pentru butoane, care să nu transmită în mod repetat aceași stare dacă butonul este ținut apăsat, dar nu am făcut asta. De ce? Nu știu, dar oricum e târziu și nu voi modifica sau ceva.

Rezultate Obţinute

Like aici pentru binecuvântări:





Concluzii

În sfârșit mi-am folosit banda de legat cabluri pe care am luat-o de pe Temu anul trecut!!

Am învățat:

- Cum se realizează pairing-ul între două module Bluetooth
- Este exponențial mai ușor controlul a două motoare, față de 4 motoare
- Să lipesc fire cu pistolul de lipit fără să mă ard tare
- Motorul cu reductor chiar consumă mult, nu mă așteptam să am nevoie de atâtea baterii
- Să nu am încredere în niciun site că îmi vinde piese legite deoarece chinezăria poate să vină și de la cei mai buni prieteni (la tine mă uit, optimus)
- Să nu renunț

Download

dreamcar.zip

Jurnal

L-am început și l-am terminat.

Bibliografie/Resurse

Serial Input Basics
Documentația Arduino IDE
ASCII TABLE
Saravanan AL, cel mai mișto indian
Muhammad Ansar, al doilea cel mai mișto indian
Behind a DC Motor
CoolTerm