STEMKIT 4SCHOOLS

CURRICULUM STEMKIT DESIGN & DEZVOLTARE

Output: O2A1

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission.

Tabel Cuprins

1 Introducere – Despre proiect

Copiii din zilele noastre se nasc în tehnologie și utilizarea lor este naturală pentru ei. Cu toate acestea, este necesar ca aceștia să dobândească abilități tehnologice, cum ar fi programarea. Forța de muncă calificată STEM are o mare cerere în Europa, iar cererea va continua să crească datorită apariției industriei 4.0 și a tehnologiilor avansate de fabricație. Sunt necesare noi modalități de implicare a copiilor în programare și STEM, dar mai mult timp petrecut pe ecran nu este cea mai bună abordare. Jocul practic este mai distractiv și de multe ori mai educativ.
Conectarea lumilor online și offline poate oferi copiilor un mediu mai captivant și mai sănătos pentru a învăța cum să programeze și să dezvolte abilități STEM.

Proiectul STEMKIT4Schools Erasmus+ are atunci ca obiectiv primordial să producă abordări și instrumente care să îi ajute pe cei care lucrează cu copii să se angajeze în programare și să își dezvolte abilitățile legate de STEM. Acesta își propune să realizeze acest lucru nu prin creșterea timpului pe ecranare, ci prin încurajarea jocului practic, prin crearea de jocuri care pot fi redate pe un computer DIY din lemn retro, în combinație cu gadget-uri electronice legate de subiectele STEM.

2 Obiectivele curriculum-ului

Un curriculum complet a fost conceput de către partenerii proiectului pentru a răspunde obiectivelor STEMKIT4Schools, inclusiv planuri de lecții pentru utilizarea Minecraft Pi / Scratch / Python / Kits - computerul STEMKIT în clasă. Planurile de lecție fac parte din ghidul educatorului pentru profesori. Kituri electronice sunt proiectate și construite pentru a completa predarea de programare, calcul fizic și subiectele STEM.

Cursul final STEMKIT predă elementele de bază ale jocurilor cu Minecraft Pi, Python, Scratch, precum și calcul fizic (folosind componente digitale, analogice și electromecanice de bază) și colaborarea (interacțiune și partajare cu ceilalți).

Computerul DIY STEMKIT se bazează pe Raspberry Pi, ceea ce înseamnă că poate valorifica puterea și resursele practic nelimitate pentru Raspberry, în timp ce caracteristicile sistemului de operare Raspbian oferă capacități suplimentare.

Curriculumul STEMKIT refăcut, utilizând principiile de proiectare a instrucțiunilor, poate fi găsit în cadrul portalului de învățare sub formă de produse de învățare interactive și va fi livrat cursanților care vor putea urmări cursul interactiv în ritmul lor, în timp ce pot utiliza instrumentele de învățare socială oferite pentru a interacționa cu colegii lor (interacționați cu comentarii, forumuri, chat). De asemenea, vor fi furnizate funcții suplimentare de colaborare, cum ar fi notificări, fluxuri de grup, bloguri, partajare de fișiere, întrebări / răspunsuri, votare.

În secțiunile următoare, sunt prezentate următoarele 5 elemente: Introducere în Scratch 2.0 [P6-ARC], Scratch GPIO (Controlează pinii GPIO / intrări de recepție) [P3-DANMAR], Introducere în Raspberry Pi Edition of Minecraft [P4-HESO / P5-SCHOLE], programare Raspberry Pi GPIO utilizând Python [P2-AKNOW] și Physical Computing [P1-ECAM].

3 Introducere în Scratch 2.0 [P6-ARC]

3.1 Glosar de termeni

Term / Concept	Definitie / Explicatii
Scratch 2.0	Scratch este un limbaj de programare, creat de MIT Media Lab, un mediu de dezvoltare open-source care facilitează crearea de artă interactivă, povești, simulări și jocuri. Acesta are ca scop educarea persoanelor cu puțină sau deloc experiență de programare, în primul rând copii cu vârste cuprinse între 8 și 16 ani. Scratch 2.0, cunoscut și ca Scratch 2, [1] a fost a doua versiune majoră a Scratch, după Scratch 1.4. A inclus un editor și un site web reproiectat și a fost prima versiune care a inclus editorul online, precum și unul offline. Scratch a fost complet rescris în Adobe Flash pentru versiunea 2.0, dar a rulat totuși proiecte din versiuni mai vechi ale Scratch. Era încă complet gratuit și fără reclame.
Blocks (2.0)	Blocurile erau forme de piese de puzzle care erau folosite pentru a crea codul în Scratch. Blocurile conectate între ele sunt asemănătoare cu un puzzle, în care fiecare tip de date (pălărie, stivă, reporter, boolean sau capac) au propria formă și un slot special format pentru a fi introdus, pentru prevenrea erorilor de sintaxă. Seria de blocuri conectate a fost numită scripturi.
Editarea culorilor blocului	În editorul Scratch 2.0 online, făcând clic cu butonul Shift pe meniul Editare a apărut o opțiune numită „Editare culori bloc”. Selectând acest lucru, va apărea un meniu cu 3 glisoare HSL și instrumente pentru modificarea culorilor de bloc ale unei categorii de blocuri specifice.

3.2 Conținut

Scratch este un **limbaj de programare vizual** bazat pe blocuri și un site web destinat în principal copiilor, care permite utilizatorilor să învețe programarea pe computer în timp ce lucrează la proiecte semnificative personal, cum ar fi povești animate și jocuri. (Maloney, 2010)

Scratch este folosit în școli de mai multe discipline (matematică, informatică, arte lingvistice, studii sociale).

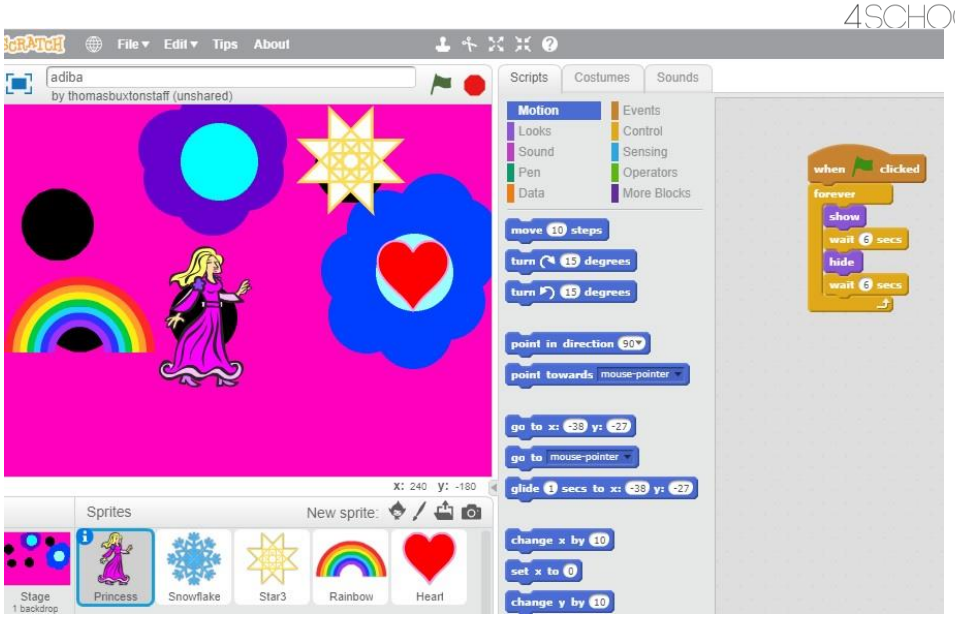


Fig.1 Captură de ecran din proiectul Scratch (Sursa https://www.thomasbuxton.towerhamlets.sch.uk/blogs/year3/2017/11/17/year-3-computing-scratch-projects/)

Proiectarea inițială Scratch a fost motivată de nevoile și interesele tinerilor (cu vârste cuprinse între 8 și 16 ani) din centrele de calcul post-școlare, cum ar fi Intel Computer Clubhouses (Resnick, 2003). Inițial, Scratch a fost utilizat în principal în medii de învățare informale, cum ar fi centrele comunitare, cluburile post-școlare, bibliotecile și casele; este folosit și în școli. (*http://web.media.mit.edu/~jmaloney/papers/ScratchLangAndEnvironment.pdf*)

3.2.1 Introducere în Scratch 2.0 - funcții de bază

Proiectul Scratch a început în 2003, iar software-ul Scratch a fost lansat public în 2007. Scratch este gratuit, disponibil în aproape 50 de limbi, software-ul este adesea redistribuit de sistemele școlare și de organizațiile educaționale.

- ✓ Un obiectiv important al Scratch este de a introduce programarea celor care nu au experiență anterioară în programare.
- ✓ Scratch vizează utilizatorii mai tineri decât celelalte două sisteme, se concentrează pe învățarea auto-direcționată, include instrumente pentru a desena imagini și a înregistra sunete.
- ✓ Scratch se bazează pe ideile constructioniste ale Logo-ului (Kay 2010; Steinmetz 2002), pentru a ajuta utilizatorii să-și facă proiectele personale atractive, motivante și semnificative.
- ✓ Scratch facilitează importul sau crearea multor tipuri de suporturi (imagini, sunete, muzică); a fost conceput pentru a invita scriptarea, pentru a oferi feedback imediat pentru executarea scriptului și pentru a face vizibile executarea și datele.
- ✓ Interfața cu utilizatorul Scratch se străduiește să faciliteze navigarea.
- ✓ Scratch-ul poate fi modificat, deoarece permite utilizatorilor să experimenteze cu comenzi și fragmente de cod așa cum s-ar putea schimba componentele mecanice sau electronice.

- ✓ Scratch nu necesită ca utilizatorul să creeze scripturi complete înainte de a rula proiectele. Fragmentele de program pot fi lăsate în panoul de scriptare și sunt salvate odată cu proiectul. (Maloney, 2010)

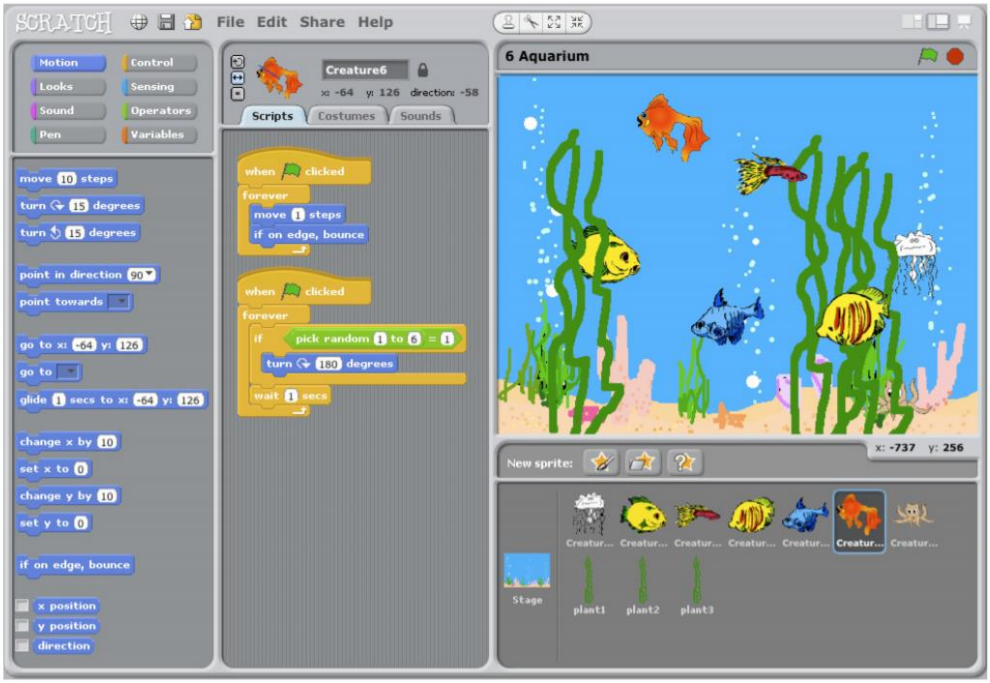


Fig. 2. Interfața utilizatorului Scratch, (Maloney, 2010)

- ✓ Scratch oferă feedback vizual pentru a arăta execuția scriptului.
- ✓ Scratch poate afișa, de asemenea, secvențierea comenzilor și fluxul de control. Activarea cu un singur pas (selectată dintr-un meniu) face ca blocurile să clipească pe măsură ce rulează.
- ✓ Folosește o singură fereastră, design cu mai multe pagini pentru a se asigura că componentele cheie sunt întotdeauna vizibile.
- ✓ Scratch evită paletele plutitoare; panoul din stânga este paleta de comenzi cu butoane pentru selectarea categoriilor, panoul din mijloc afișează scripturile pentru sprite-ul selectat în prezent, cu file de folder pentru a vizualiza și edita costumele (imaginile) și sunetele deținute de acel sprite. Panoul mare din dreapta sus este scena, unde se întâmplă acțiunea. Panoul din dreapta jos arată miniaturile tuturor spriturilor din proiect, cu spritul selectat în prezent evidențiat.
- ✓ Pentru a invita scripturi, paleta de comenzi este întotdeauna vizibilă. Comenzile sunt împărțite în opt categorii, cum ar fi Motion, Looks, Sound și Control. Acest lucru evită listele lungi, potențial copleșitoare, de comenzi: în majoritatea paletelor, toate comenzile pot fi vizualizate fără derulare. În fiecare categorie, comenzile cele mai auto-explicative și utile apar în partea de sus a paletei de comenzi. Blocurile de comandă sunt codificate în culori pe categorii, ajutând utilizatorii să găsească blocuri conexe..

(Sursa: http://web.media.mit.edu/~jmaloney/papers/ScratchLangAndEnvironment.pdf)

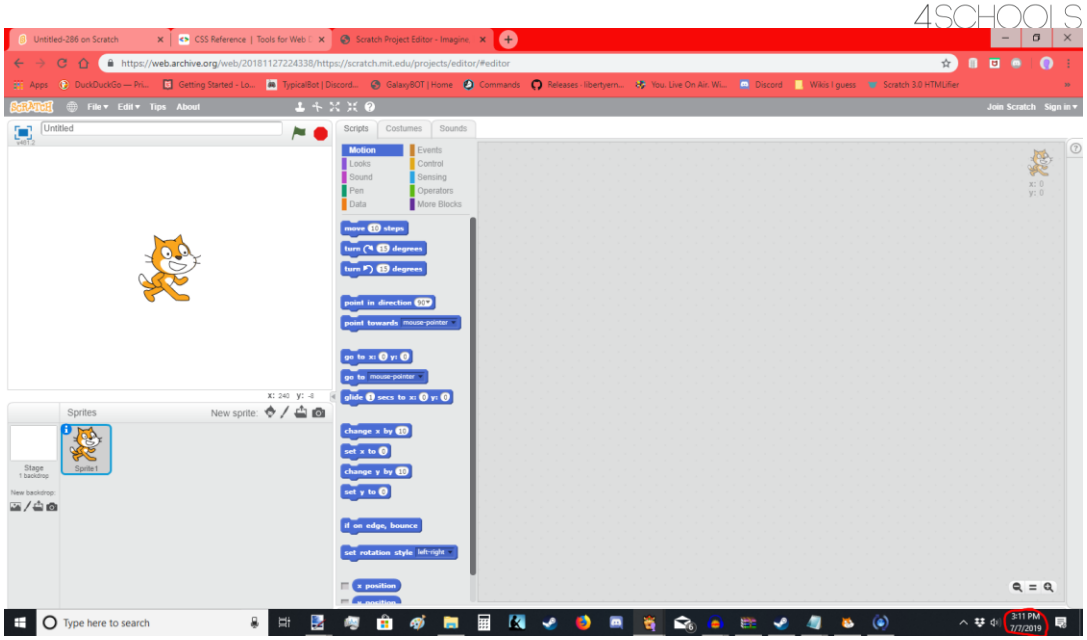


Fig. 3. Scratch 2.0 editor - online (https://scratch.mit.edu/)

Scratch actualizează afișajul după fiecare comandă. Vederea efectului fiecărei comenzi, chiar dacă este doar ca o scurtă bliț pe ecran, oferă indicii vizuale importante la depanare. Scratch 1.0 avea 92 de blocuri de comandă. Pe măsură ce Scratch a evoluat, a fost o luptă constantă pentru a menține numărul de comenzi redus.
Au fost adăugate mai multe comenzi decât eliminate. Scratch 1.4 are 125 de blocuri de comandă, deși unele dintre ele nu apar până când este necesar.
Dacă aveți un computer mai vechi sau nu puteți instala editorul offline Scratch 2.0, puteți încerca să instalați Scratch 1.4.
Dacă sunteți administrator de rețea: un MSI Scratch 2.0 a fost creat și întreținut de un membru al comunității și găzduit pentru descărcare publică la http://llk.github.io/scratch-msi/.
Una dintre erorile din Scratch 2.0 a fost capacitatea pentru Scratchers de a urmări utilizatorii pe care nu îi pot urmări în mod normal, cum ar fi ei înșiși sau utilizatorii șterși.

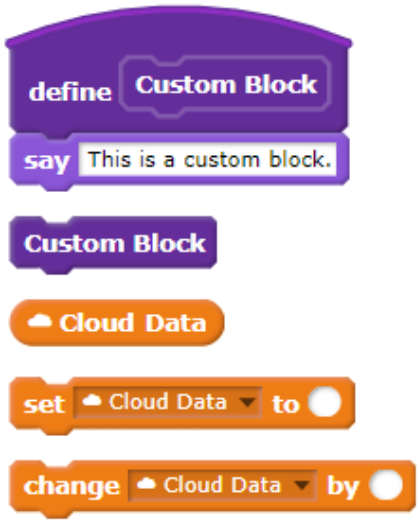


Fig. 4 Exemple de New Features și Blocks

Scratch permite conectarea blocurilor numai în moduri semnificative. Un bloc de comandă se conectează atunci când este lăsat în secvența de comandă, dar un bloc funcțional nu se va conecta dacă este lăsat în același loc. Pe măsură ce utilizatorul trage un bloc, Scratch oferă feedback vizual care arată posibile puncte de inserare a secvenței (blocuri de comandă) sau ținte de sloturi de parametri (blocuri funcționale). (Maloney, 2010)

Table 1. Scratch Block Types (Maloney, 2010)

	A *command* block has a notch on the top and a matching bump on the bottom. Command blocks can be joined to create a sequence of commands called a *stack*.
	A *function* block returns a value. Function blocks do not have notches.
	A *trigger block* has a rounded top. It runs the tack below it when the triggering event occurs.
	*Control structure* command blocks have openings to hold nested command sequences.

În Scratch, un bloc de structură de control este o unitate indivizibilă. Brațul de închidere al unei bucle sau blocuri condiționale face parte din blocul în sine - nu poate fi deplasat greșit - și cuibărirea secvenței de comandă închise este manifestă.

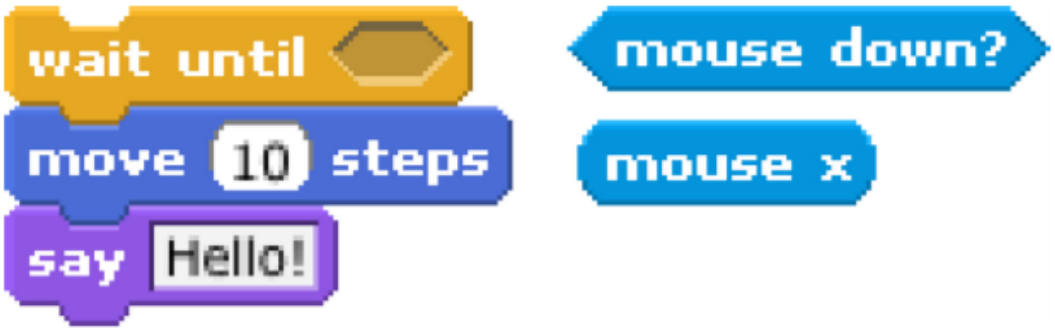


Fig. 5. Formele indică tipul. În stânga, blocuri de comandă cu sloturi de parametrii booleani, numerici și șiruri. În dreapta, blocurile funcționale booleane și numerice. (Maloney, 2010)

3.2.2 Exemple Practice - proiecte Scratch

Limbajul blocurilor Scratch elimină erorile de sintaxă, permițând utilizatorilor să se concentreze imediat asupra problemelor interesante. Mesajele de eroare în timpul rulării sunt evitate prin comenzi failsoft, în timp ce un model de concurență atent conceput, evită condițiile de cursă.
Pentru fiecare activitate educațională, puteți încerca Tutorialul, puteți descărca un set de carduri de codare, sau puteți vizualiza Ghidurile pentru educatori.

Limbajul de programare Scratch subliniază simplitatea. Sistemul de tip și modelul de obiecte au fost concepute pentru a funcționa fără probleme, fără a fi explicate în prealabil, dar pentru a avea un sens perfect la o examinare mai atentă.

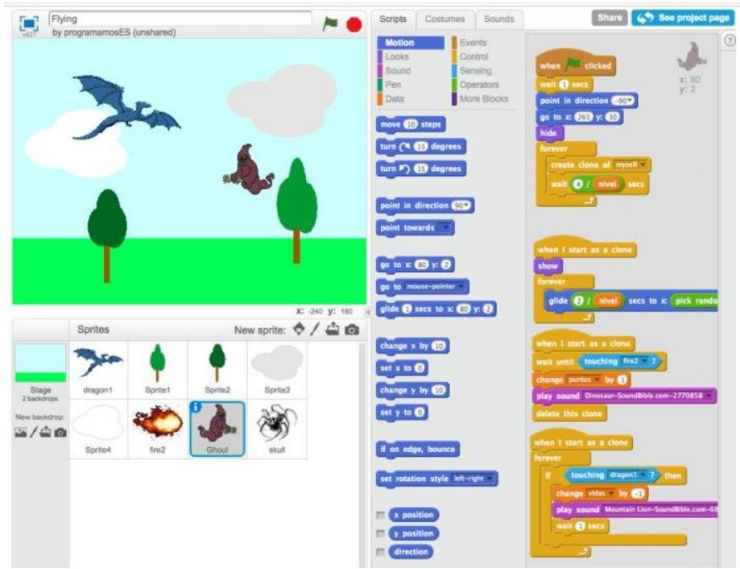


Fig. 6. Screenshot - Exemple dintr-un proiect Scratch. În dreapta, elementele vizuale utilizate pentru a programa în mediul Scratch.
(*ComputerProgrammingInTheEnglishClassroom.pdf)*

Structura interfeței Scratch ne face mai ușor să jucăm și să explorăm idei. Pentru a crea un joc, o poveste sau o animație în Scratch, stivuim blocuri împreună pentru a forma un script care oferă instrucțiuni sprite-urilor proiectului.

Pe măsură ce creăm proiecte, ne evaluăm munca și determinăm dacă rezultatele îndeplinesc așteptările noastre. Este foarte ușor, deoarece totul se întâmplă într-o singură interfață.

Activitate - Cardurile de salut (https://scratch.mit.edu/projects/11739928)
Educatorii pot remixia Idei:
- Editați cardul pentru o altă ocazie
- Schimbați imaginile pentru a se potrivi cu o temă
- Redați o animație în interiorul cardului

Elevii își pot încorpora vocile și imaginile într-un proiect, creând ceva care îi ajută pe ceilalți să învețe despre ei înșiși și despre oameni, problemele și lucrurile de care țin. Cu Cardurile de codare Scratch, elevii pot învăța să creeze jocuri interactive, povești, muzică, animații și multe altele!
Jucătorul ar trebui să aibă un script care să-și controleze mișcarea folosind indicatorul mouse-ului. De asemenea, ținta ar trebui să fie programată pentru a se afișa în locuri aleatorii. Dacă designul jocului dvs. este diferit, poate fi necesar să ajustați sistemul de notare pentru a se potrivi proiectului dvs. Scratch.

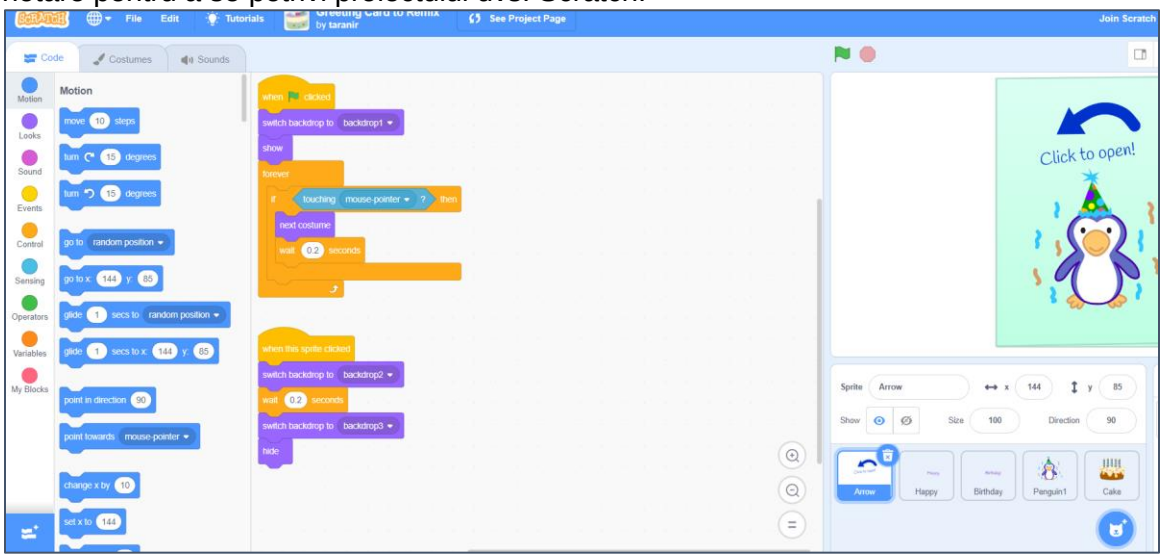


Fig. 7. Exemple screenshot - proiect Scratch,
(https://scratch.mit.edu/projects/11739928/editor/)

Scriptul pe care îl veți construi necesită utilizarea unei *variabile.* O variabilă este un factor care se poate schimba. Veți crea o variabilă de scor. Acesta va fi folosit pentru a deține numărul de puncte.
Un bloc de codare va fi folosit pentru a crește scorul cu un număr specificat de puncte.

Un alt bloc de codare va reseta variabila de scor la zero când începe un joc nou. 

Mediul de programare Scratch și limbajul funcționează împreună pentru a crea un sistem care este extrem de rapid de învățat, menținând utilizatorii implicați ani de zile; utilizatorii pot programa în decurs de cincisprezece minute.

3.2.3 Concluzii

Scratch are un număr mic de comenzi, permite schimbul de sprite fără a sparge dependențe, favorizând colaborarea și partajarea codului. Sistemul este întotdeauna activ, fără comutator de rulare / editare, astfel încât comenzile sau fragmentele de cod pot fi executate cu un clic, iar feedback-ul grafic arată execuția. Variabilele și listele au vizualizări concrete, astfel încât efectul operațiilor de date poate fi văzut imediat. (Maloney, 2010)

Capacitatea de a codifica programe de calculator este o parte importantă a alfabetizării în societatea actuală. Când oamenii învață să codeze în Scratch, învață strategii importante pentru rezolvarea problemelor, proiectarea proiectelor și comunicarea ideilor.
Proiecte Scratch - bazate pe unități de învățare vor include diferite discipline și această nouă perspectivă va permite elevilor să aplice ceea ce învață în situații noi, ducând la o învățare mai profundă; elevii vor fi implicați în activități de proiectare, urmărind interese personale, interacționând prin colaborări creative și reflectând asupra experiențelor utile.

3.3 Test de evaluare

1. Scratch nu este doar pentru orele de informatică. Scratch poate fi încorporat în orice zonă de conținut din orice clasă.
    1. **DA**
    2. NU

2. Scratch este disponibil atât online, cât și ca fișier descărcabil.
    1. **DA**
    2. No

3. Aducându-i pe elevi să creeze un joc de matematică cu programul lor de zgârieturi, ei pot apoi să aplice ceea ce au învățat în lecțiile lor într-o activitate practică.
    1. **DA**
    2. NU

4. Elevii își pot încorpora vocile și imaginile într-un proiect Scratch.
    1. **DA**
    2. NU
5. Puteți ține scorul în Scratch creând un ...
    1. Sprite
    2. Loop
    3. **Variabile**

4. Functia

6. Mai multe declarații unite sunt un ...
   1. Run
   2. **Sequence**
   3. Loop
   4. Variable

7. Cum se creează a loop in scratch?
   1. **Folosiți un bloc de repetare**
   2. Fixați un bloc
   3. Folosiți un bloc de condiții
   4. Folosiți o variabilă

8. Adevărat sau Fals: puteți adăuga propriile imagini în proiectele dvs. Scratch:
   1. **Adevărat**
   2. Fals

9. Efectuarea unei acțiuni în funcție de IF este îndeplinit un criteriu se numește a ..an action depending on IF a criterion is met is called a ..
   1. secvență
   2. declarație
   3. bucla
   4. **stare**

10. Repetarea unei declarații de mai multe ori se numește..
   1. **Loop**
   2. Repetiție
   3. Eveniment
   4. Secvență

3.4 Referințe

- KAY, A. 2010. Squeak etoys, children, and learning. http://www.squeakland.org/resources/articles
- Resnick, M., Maloney, j., Monroy-Hernandez, 2009. Scratch: Programming for all. Comm. ACM 52, 11, 60–67.
- ComputerProgrammingInTheEnglishClassroom.pdf
- Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E. 2010. *The scratch programming language and environment*. ACM Trans. Comput. Educ. 10, 4, Article 16 (November 2010), 15 pages. DOI = 10.1145/1868358.1868363. http://doi.acm.org/10.1145/1868358.1868363
- https://education.abc.net.au/home#!/media/1214681/intro-to-scratch-20
- https://scratch.mit.edu
- http://web.media.mit.edu/~jmaloney/papers/ScratchLangAndEnvironment.pdf
- https://www.thomasbuxton.towerhamlets.sch.uk/blogs/year3/2017/11/17/year-3-computing-scratch-projects/

4 Scratch GPIO (Control GPIO pins/receive inputs) [P3-DANMAR]

4.1 Glosar de termeni

Termen / Concept	Definitie / ExplicaȚIE
Pi GPIO extension	Extensia GPIO Pi este o componentă suplimentară pentru Scratch care permite interacțiunea cu pinii GPIO ai Raspberry. Această extensie oferă două blocuri suplimentare pentru a citi și seta starea tuturor celor 28 de pini GPIO furnizați de Raspberry.
Raspbian	Raspbian este un sistem de operare bazat pe Debian care este optimizat pentru a fi utilizat pe dispozitivele Raspberry. Oferă o interfață GUI simplă și este inclusă în instalarea Scratch și alte instrumente de programare care permit utilizatorilor să interacționeze cu hardware-ul Raspberry.

4.2 Conținut

Scratch este adesea considerat ca o intrare în lumea programării. Algoritmi simpli pot fi dezvoltați folosind acest software, iar stratul de prezentare compus din multe imagini este atrăgător. Cu toate acestea, Scratch pentru aplicații mai avansate și mai interesante poate părea puțin limitat. Din fericire, suportul GPIO care poate fi activat permite utilizarea pinilor Raspberry pentru a controla mai multe circuite fizice, dispozitive și aparate diferite.

4.2.1 Informații de bază despre Scratch GPIO

Mediul Scratch este excelent pentru cei care încep cu programarea microcontrolerului. Cu utilizarea Scratch, este posibil nu numai să proiectăm algoritmi specifici și să le rulăm, ci și să controlăm porturile GPIO ale Raspberry pentru a îmbunătăți funcționalitatea întregului sistem. Scratch acceptă GPIO din cutie, iar acest lucru se poate face prin adăugarea unei extensii GPIO speciale Pi.
Odată ce extensia GPIO este adăugată la Scratch, două blocuri noi sunt disponibile pentru utilizare ulterioară. Acestea sunt: blocul stivă și blocul boolean. Blocul de stivă este o instrucțiune simplă care poate fi utilizată pentru a seta orice port GPIO la starea ridicată sau scăzută. Blocul boolean, pe de altă parte, poate fi utilizat pentru a verifica dacă vreun pin GPIO dat este în stare înaltă sau joasă. Acest lucru permite Scratch să controleze toți pinii GPIO disponibili ai plăcii Raspberry.

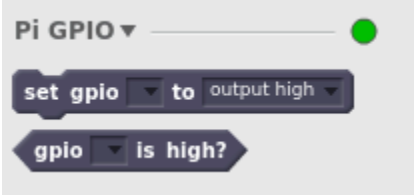


FIGURA 1 BLOCURI SUPLIMENTARE FURNIZATE DE EXTINDEREA GPIO

În general vorbind, extensia GPIO pentru Scratch deschide o gamă largă de noi posibilități, deoarece utilizatorul nu mai este restricționat în proiectarea algoritmilor din

software-ul Scratch, ci este în schimb capabil să interacționeze cu dispozitive fizice, cum ar fi diode, buzzere și așa mai departe. Blocurile furnizate de extensie sunt pe deplin compatibile cu blocurile Scratch standard (încorporate), extinzând astfel și mai mult zona largă de opțiuni.

Pentru a adăuga extensia GPIO la Scratch, este necesar să deschideți aplicația și din fila Scripturi selectați ultima opțiune Mai multe blocuri. După ce se face clic pe butonul Mai multe blocuri, este necesar să faceți clic pe Adăugați o extensie.

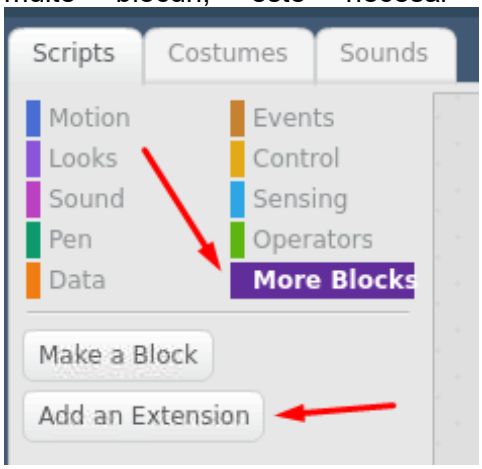


FIGURA 2 ACTIVAREA EXTENSIEI GPIO

Când este afișat ecranul cu Extension Library, ar trebui făcută selecția extensiei Pi GPIO și alegerea trebuie confirmată.

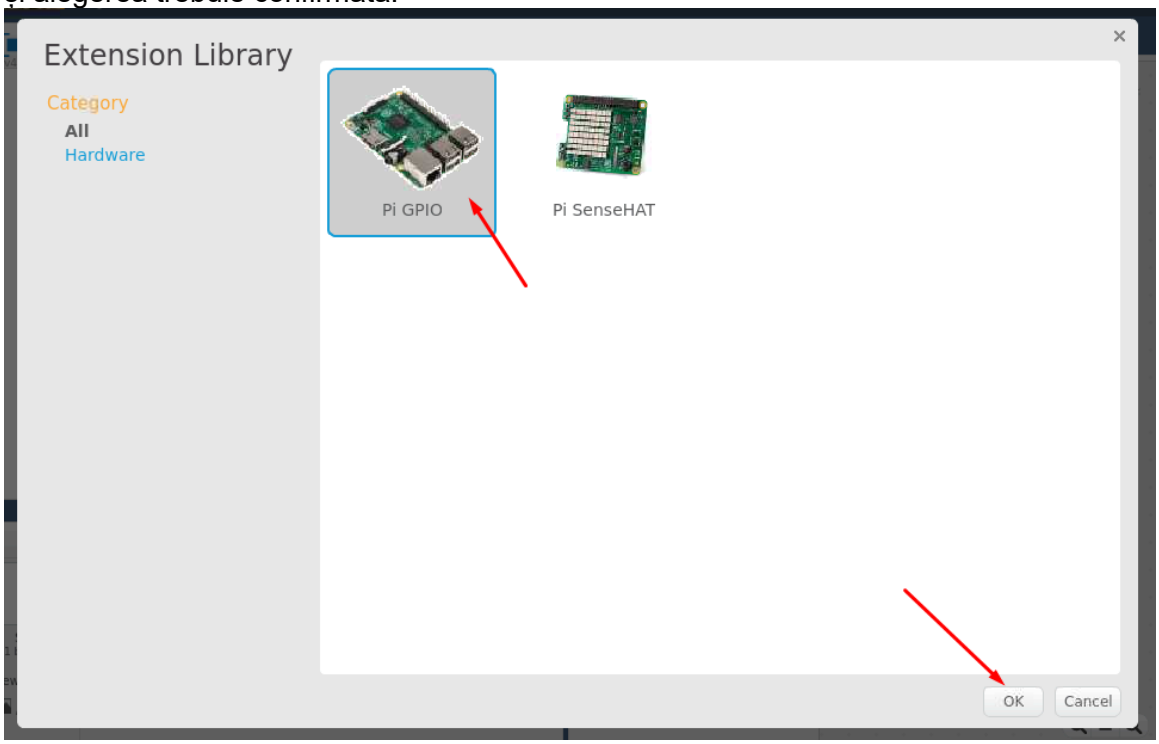


FIGURE 3 ENABLING GPIO EXTENSION

După parcurgerea pașilor de mai sus, Scratch este gata să controleze GPIO-ul Raspberry.

Pentru a asigura o experiență perfectă, cel mai bun mod de a pune în funcțiune mediul, este recomandabil să instalați un sistem de operare Raspbian pe Raspberry, deoarece această distribuție are totul deja configurat. Sunt disponibile diferite versiuni Scratch, astfel încât utilizatorii să o poată selecta pe cea cu care sunt cel mai familiarizați.

4.2.2 GPIO convenție de numire

Este important de menționat că numărul pinilor GPIO utilizați în Scratch respectă convenția de denumire utilizată de Raspberry în sine. Acest lucru poate fi confuz, în special pentru începători, deoarece Raspberry în majoritatea modelelor folosește un conector cu 40 de pini, iar pinii sunt numerotați de la 1 la 40. Pentru a evita confuzia, cel mai bine este să aveți așa-numita diagramă pinout pentru Raspberry. Un exemplu disponibil în documentația oficială Raspberry este prezentat mai jos.

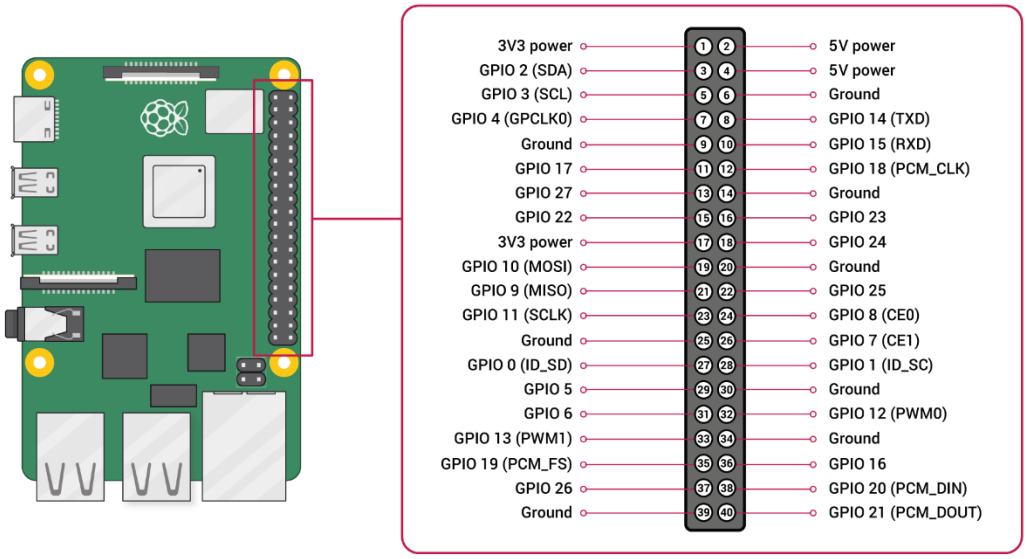


FIGURA 4 RASPBERRY PINOUT CHART

Prin urmare, dacă codul din Scratch folosește numărul 26, acesta indică pinul fizic 37 (coloana din stânga, a doua din partea de jos). Este important să vă familiarizați cu locația pinilor GPIO specifici pentru a vă asigura că nu există confuzie între numerele care indică pinii fizici și cei care indică pinii GPIO..

4.2.3 GPIO interactiune

După cum sa menționat deja, blocurile GPIO sunt pe deplin compatibile cu blocurile furnizate de Scratch. Prin urmare, cel mai simplu exemplu pentru a demonstra modul în care pini specifici pot fi controlați de la Scratch, o simulare a fulgerului și a unui LED poate fi făcută într-un mod foarte de bază. Următorul presupune că LED-ul este conectat la pinul GPIO 11.

FIGURA 5 UTILIZAREA UNUI PIN GPIO DE IEȘIRE ÎN SCRATCH

Exemplul de mai sus folosește blocul de stivă furnizat de extensia GPIO. Blocul boolean poate fi, de asemenea, utilizat, de exemplu, pentru a acționa la detectarea unei stări ridicate pe orice pin GPIO dat.

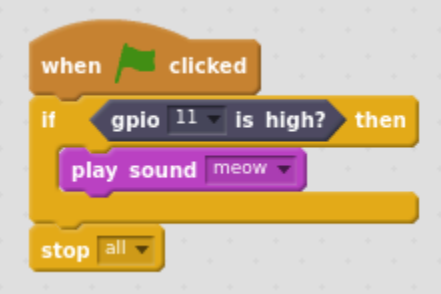


FIGURA 6 FOLOSIREA UNUI PIN GPIO DE INTRARE ÎN SCRATCH

S-ar putea părea că a avea doar două blocuri suplimentare este oarecum limitativ. În realitate, însă, programarea microcontrolerului este vorba despre două stări: înaltă și joasă. Mai mult, toate dispozitivele digitale moderne funcționează astfel. Stările înalte și joase pot fi traduse în valori logice de 0 și 1 și, în realitate, acest lucru înseamnă că nu există tensiune (0V) sau tensiune (3,3V), respectiv. Pentru Raspberry, GPIO funcționează la 3,3 V și este bine să țineți cont de acest lucru pentru conectarea diferitelor dispozitive, senzori și echipamente la acești pini.
Extensia GPIO poate fi, de asemenea, un înlocuitor interesant pentru proiectele Scratch existente. În loc să interacționeze cu utilizatorul numai cu utilizarea ecranului Scratch, unele operații pot fi transferate pe pinii GPIO fizici. De exemplu, în loc să afișeze un mesaj sau să redea un sunet, extensia GPIO poate fi utilizată pentru a emite sunetul, a porni LED-ul și așa mai departe. Posibilitățile acestei configurări sunt limitate doar de imaginația și creativitatea unei persoane care lucrează la un anumit proiect folosind Scratch cu suportul GPIO al Raspberry.

4.3 Practice exemple

În următoarele câteva pagini sunt oferite câteva exemple practice. Aceste exemple sunt destul de ușor de creat și scopul lor principal este de a demonstra modul în care Scratch poate fi utilizat pentru a controla pinii GPIO pe Raspberry.

4.3.1 LED intermitent – exemplul 1

Primul exemplu practic va introduce noțiunile de bază ale lucrului cu GPIO controlat de Scratch. Ideea este că va fi folosit un circuit foarte simplu care va clipi LED-ul. LED-ul va fi controlat de pinul GPIO numărul 11 (pinul fizic 23). În timp ce poate fi utilizat orice alt

GPIO, pinul GPIO 11 este foarte convenabil de utilizat, deoarece are masă (GND) lângă acesta (pinul fizic 25).
Pentru a configura acest circuit, este nevoie de un LED, un rezistor potrivit pentru LED-uri, o placă de măsurare și două fire jumper.
Zmeura ar trebui să fie conectată la panoul de măsurare așa cum se indică mai jos.

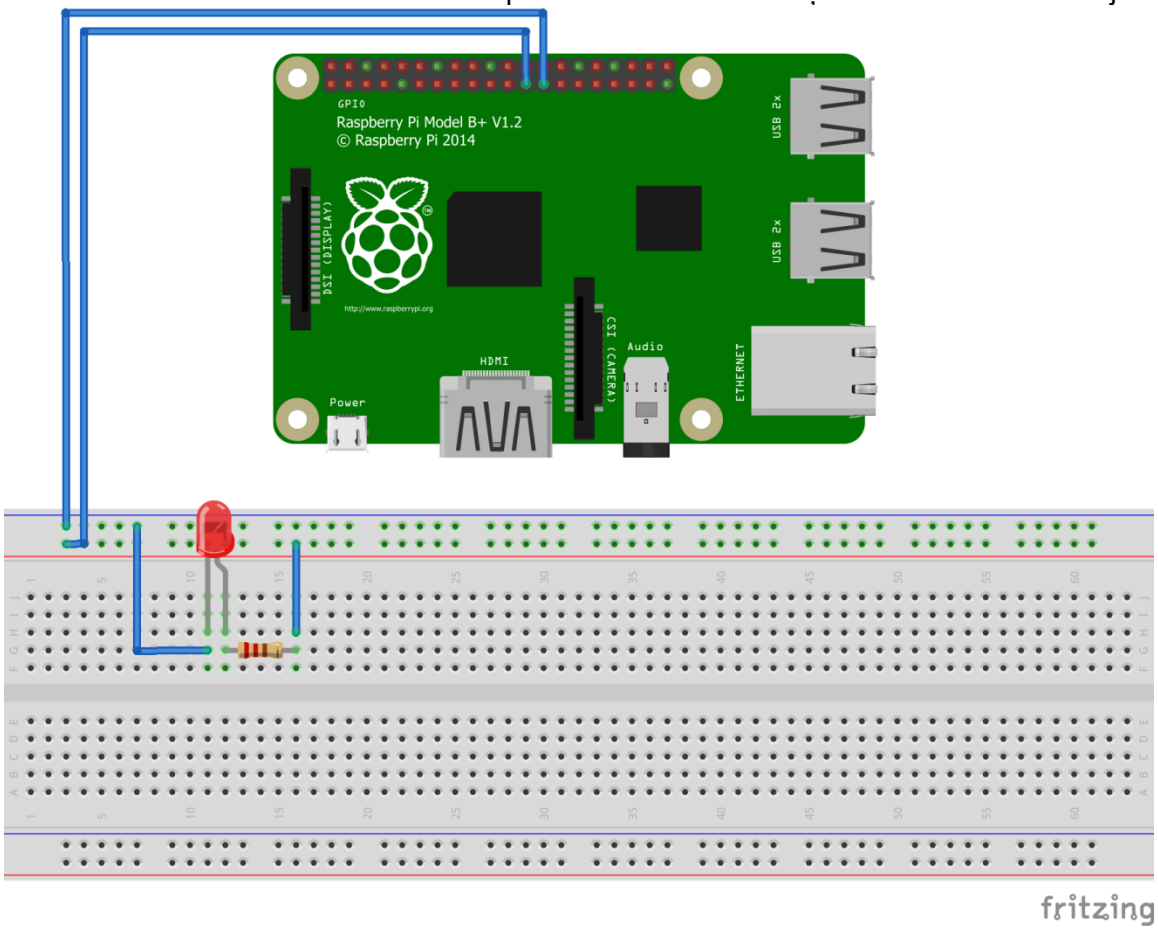


FIGURA 7 CONECTIAREA LED LA RASPBERRY

Pentru a face acest circuit foarte simplu să funcționeze, pinul GPIO selectat 11 ar trebui să fie controlat de Scratch care își va seta starea la înaltă, va întrerupe o secundă, va seta starea la scăzută și va întrerupe din nou o secundă. Aceste patru instrucțiuni pot fi plasate într-o buclă pentru totdeauna, dar pot fi utilizate și alte tipuri de bucle, cum ar fi repetarea. Setul eșantion de blocuri este demonstrat mai jos pentru referință.

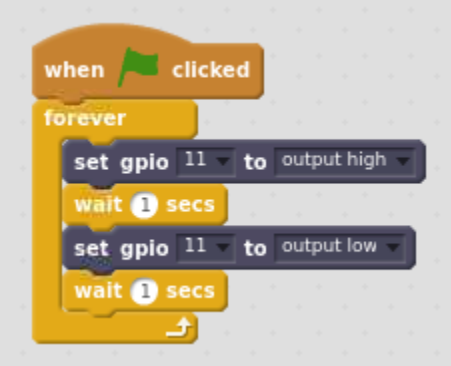


FIGURA 8 COMUTAREA STĂRII PIN GPIO ÎN SCRATCH

Ceea ce se întâmplă aici este că Scratch comunică cu Raspberry folosind extensia GPIO și emite instrucțiuni care permit schimbarea stării pinului de ieșire (11) de la mare la scăzut. Acest lucru se traduce prin faptul că tensiunea este prezentă pe pinul GPIO în intervale de timp specifice.
Deși acest exemplu este foarte simplu, acesta oferă un punct de plecare pentru controlul GPIO-ului Raspberry cu utilizarea Scratch. În acest circuit simplu, pinul GPIO 11 acționează ca un pin de ieșire, ceea ce înseamnă că primește doar instrucțiuni pentru a-și schimba starea în consecință. Cu extensia GPIO este, de asemenea, posibil să citiți starea pinului GPIO care va acționa ca un pin de intrare. Această abordare este prezentată în exemplul următor.

4.3.2 LED de control cu un comutator tactil - exemplu 2

Pentru a prezenta modul în care Scratch poate citi pinul GPIO al Raspberry ca intrare, poate fi construit un circuit simplu. În plus față de exemplul prezentat mai sus, este necesar un comutator tactil suplimentar, împreună cu un rezistor de tragere de 10 k și fire de jumper suplimentare.
Circuitul care a fost utilizat într-un exemplu anterior trebuie extins printr-un comutator tactil care va seta pinul GPIO la o stare înaltă odată apăsat. Pentru a face acest lucru, poate fi utilizat pinul GPIO numărul 26 (pinul fizic 37).
Deoarece comportamentul așteptat este că pinul GPIO 26 citește starea scăzută atunci când comutatorul tactil nu este apăsat și ridicat în caz contrar, este necesar să fie plasat un rezistor pull-down între GND și pinul GPIO 26. Datorită acestui fapt, ori de câte ori este emisă cererea de citire a stării acului, acesta va citi scăzut (0V) atunci când comutatorul tactil nu este apăsat. Pe de altă parte, când este apăsat comutatorul tactil, citirea ar trebui să fie ridicată. Acest lucru poate fi realizat prin conectarea comutatorului tactil între pinul GPIO 26 și linia 3,3V furnizat de Raspberry pe pinii fizici 1 și 17 care sunt deja furnizați cu fire jumper la o placă de măsurare. Apăsarea comutatorului tactil va face ca curentul să curgă de pe o linie de 3,3V, astfel setând pinul GPIO 26 într-o stare înaltă. Un exemplu

de circuit este furnizat mai jos pentru referință.

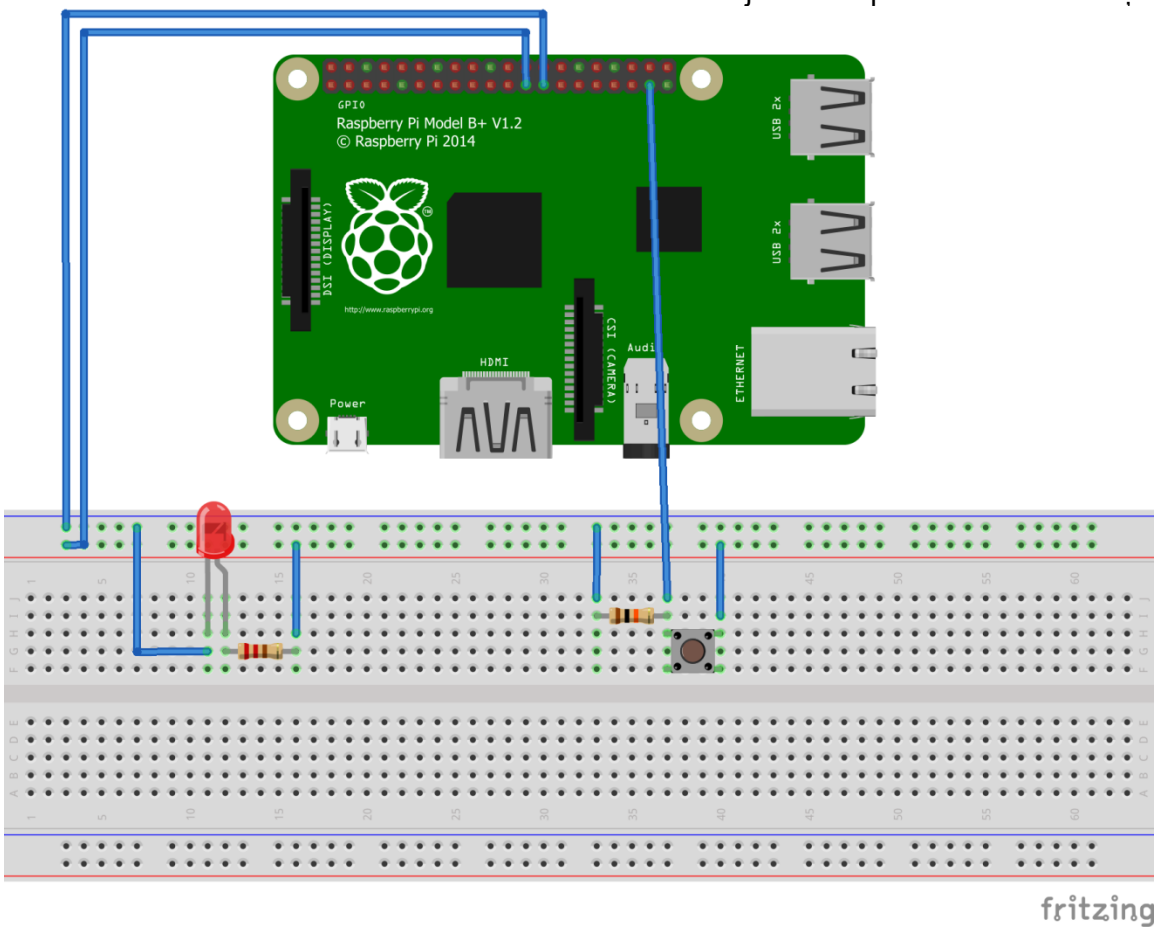


FIGURA 9 CIRCUIT LED DE CONTROL AL COMUTATORULUI TACTILE

În cele din urmă, codul Scratch poate fi furnizat pentru ca întregul circuit să funcționeze conform planificării. Pentru a face acest lucru, poate fi utilizată o buclă pentru totdeauna. În această buclă pinul GPIO 26 trebuie citit. În termenii Scratch, un bloc de stivă poate fi folosit ca parametru pentru instrucțiunea if. Restul instrucțiunilor urmează logica deja schițată a circuitului.

FIGURA 10 CITIREA STATULUI GPIO ÎN SCRATCH

De îndată ce comutatorul tactil este apăsat, pinul GPIO 26 va fi pus într-o stare înaltă. Dacă este detectată o astfel de stare, pinul GPIO 11 utilizat anterior ar trebui să fie setat la înalt pentru a porni LED-ul. Diferența dintre cei doi pini GPIO este că pinul 26 este considerat un pin de intrare (iar Scratch își verifică continuu starea), în timp ce pinul 11 este un pin de ieșire care controlează LED-ul pornind (asigurând tensiune în stare înaltă) și oprit. (fără tensiune în stare joasă).

4.3.3 Extinderea utilizării GPIO - exemplu 3

Utilizarea Scratch pentru a controla GPIO-ul Raspberry poate duce la o iluzie falsă că, în cel mai rău caz, codul nu va funcționa. Cu toate acestea, în realitate, interacțiunea cu hardware-ul Raspberry necesită o atenție suplimentară. Unul dintre exemple este că pinii GPIO nu au voie să atragă mai mult de 50mA de curent. Deși acest lucru este în regulă pentru un LED sau un semnal sonor, orice resurse mai înfometate de curent nu vor funcționa și pot duce la deteriorarea permanentă a plăcii Raspberry. Totuși, astfel de cazuri pot fi rezolvate cu ajutorul unui releu și a unei surse de alimentare externe.
Un comutator de releu funcționează de obicei la 5V și această tensiune este furnizată de Raspberry pe pinii 2 și 4. În același timp, pinul de ieșire GPIO al Raspberry setat la o stare înaltă este suficient pentru declanșarea releului și comutarea acestuia în modul de operare. Circuitul extern conectat la releu va fi închis, iar curentul va începe să curgă, alimentând orice dispozitiv. Acest lucru poate fi ilustrat prin configurarea unui circuit în care Scratch controlează releul cu ajutorul unui pin de ieșire GPIO, iar releul pornește apoi un motor care necesită (ca exemplu) 9V.
Comutatorul releului poate fi conectat la pinii 4 (ieșire 5V) și 6 (GND). Controlul / direcția releului poate fi conectat la pinul GPIO 11 care va fi folosit ca pin de ieșire. Apoi, circuitul extern (un motor de 9V cu sursă de alimentare proprie) trebuie să fie conectat la comutatorul releului. Un exemplu de configurare este furnizat mai jos pentru referință.

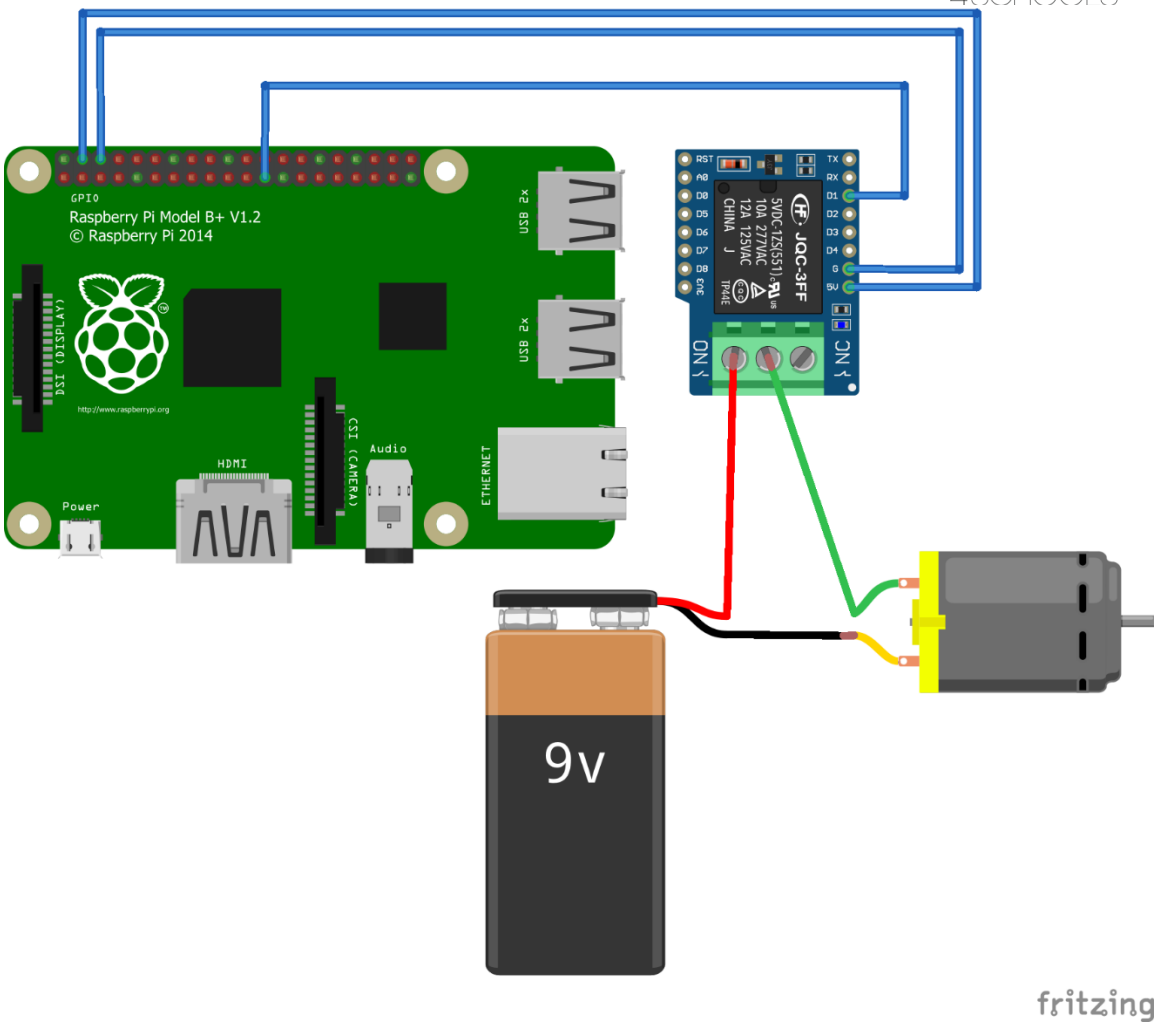


FIGURA 11 FOLOSIREA UNUI COMUTATOR DE RELEU PENTRU A PORNI MOTORUL

Inside Scratch, instrucțiunea care va activa comutatorul de releu este un bloc de stivă care va seta pinul GPIO 11 la stare înaltă. Acest lucru poate fi declanșat folosind un sprite care reprezintă pisica. O altă îmbunătățire față de exemplele anterioare este că va exista o intrare furnizată de Scratch care va acționa diferit pe baza stării curente (citire) a pinului GPIO 11.

În acest scop, poate exista o acțiune care va fi executată de fiecare dată când se face clic pe sprite-ul pisicii. Dacă citirea curentă a pinului de ieșire GPIO este mare, aceasta va fi setată la scăzută. Dacă este scăzut, acesta va fi setat la ridicat. Pinul GPIO va rămâne în starea setată, acționând astfel ca un simplu comutator în două stări. În plus, pisica va spune „Bună ziua!” pentru a avea un feedback vizual suplimentar că blocul de cod este într-adevăr executat. Un exemplu de cod Scratch este furnizat mai jos pentru referință.

FIGURA 11 SPRITE FUNCȚIONÂND CA UN INTERRUPTOR CU DOUĂ STATE CITIND STATUL SĂU

În acest exemplu s-a demonstrat că porturile GPIO ale Raspberry pot fi utilizate într-un mod destul de flexibil. Chiar dacă pinul GPIO 11 este utilizat în cea mai mare parte ca acționare a pinului de ieșire (pornirea și oprirea comutatorului releului), în același timp este posibil să construiți logica circuitului în jurul acestuia verificând starea sa curentă.

4.4 Testde evaluare

1. Scratch necesită o extensie suplimentară pentru a funcționa cu GPIO-ul Raspberry
   a. **Adevărat**
   b. Fals
2. Extensia GPIO adaugă trei blocuri suplimentare la Scratch
   a. Adevărat
   b. **Fals**
3. Dacă pinul GPIO este utilizat ca ieșire, Scratch nu este capabil să-i citească starea
   a. Adevărat
   b. **Fals**
4. Curentul maxim pe care îl extrage orice circuit conectat din portul GPIO este:
   a. 25mA
   b. 35mA
   c. **50mA**
5. Verificarea stării unui pin GPIO se poate face prin:
   a. un bloc de stive
   b. **un bloc boolean**
   c. un bloc de control
6. Extensia GPIO folosește numere care denotă:
   a. Pin-urile numerotate ale Raspberry
   b. **Numerele atribuite de Raspberry GPIO**
   c. Convenția de numire a zgârieturilorRaspberry's numbered pins
7. Setarea unui pin GPIO la starea ridicată în Scratch va duce la tensiunea pinului:
   a. 0V
   b. 1V

   c. **3.3V**
8. Scratch poate controla numai pinii GPIO selectați
   a. Adevărat
   b. **Fals**
9. Sistemul de operare sugerat pentru a funcționa cu Scratch și GPIO este:
   a. Windows
   b. **Raspbian**
   c. macOS
10. Blocul de stivă furnizat de extensia GPIO este utilizat pentru:
   a. **modificați starea pinului GPIO**
   b. citiți starea pinului GPIO
   c. toate cele de mai sus

4.5 Referințe

- https://www.raspberrypi.org/documentation/usage/gpio/
- https://www.raspberrypi.org/documentation/usage/gpio/scratch2/README.md
- https://www.circuits.dk/everything-about-raspberry-gpio/
- https://thepihut.com/blogs/raspberry-pi-tutorials/tutorial-tactile-switch
- https://raspberrypihq.com/use-a-push-button-with-raspberry-pi-gpio/

4.6 Resurse

1. Raspberry Pi GPIO: https://www.raspberrypi.org/documentation/usage/gpio/
2. Scratch Wiki: https://scratch-wiki.info/
3. An alternative GPIO extension for Scratch: https://raspberry-valley.azurewebsites.net/GPIO-with-Scratch/
4. Sparkfun education on Scratch and GPIO: https://sparkfuneducation.com/how-to/scratch-gpio-control-guide.html
5. Tutorial for beginners on Scratch and GPIO: https://www.magicbytes.com/coming-soon/gaming-computer-lab/tutorials-s/learn-to-code/scratch-gpio-beginner

5 Introducere în Raspberry Pi Edition - Minecraft [P4-HESO / P5-SCHOLE]

5.1 Glosar de termeni

Term / Concepte	Definitie / Explicatie
Raspberry Pi	Raspberry Pi este un computer cu dimensiuni de card de credit, complet funcțional, care funcționează pe Raspberry Pi OS.
Minecraft	Minecraft este un joc educațional în lume deschisă, unde jucătorii își pot construi propriile lumi virtuale cu blocuri care reprezintă materiale diferite.
Raspberry Pi OS	Sistemul de operare pentru Raspberry Pi.
Python	Limbaj de programare orientat pe obiecte care va fi folosit pentru a construi automat lucrurile în Minecraft.

5.2 Conținut

Minecraft Pi este o versiune a Minecraft, cu caracteristici minime, dezvoltată pentru Raspberry Pi. Ediția Pi este concepută ca un instrument educațional pentru programatorii începători, permițând utilizatorilor să se bucure de joc și să învețe programarea în același timp. Acest document prezintă cele mai importante și practice îndrumări pentru Minecraft Pi, cum ar fi cum să controlezi playerul, să construiești manual cu blocuri și să folosești interfața Python pentru a manipula lumea din jurul tău. Este menit în scopuri educaționale și este considerat un manual rapid, dar all-inclusive, pentru introducerea unui nou jucător în Minecraft Pi.
Modulul 3 constă din următoarele elemente:

- Introducere în funcțiile de bază Minecraft Pi
- Elemente și joc Minecraft Pi
- Controlarea Minecraft Pi cu Python
- Interacțiunea Minecraft Pi cu lumea fizică prin GPIO-ul Raspberry Pi:
    - Conectarea LED-urilor, butoanelor și comutatoarelor
    - Creați kituri electronice pentru a interacționa cu Minecraft Pi
- Importul de noi hărți și pachete de resurse Minecraft Pi
- Exemple practice de Minecraft Pi și Python
- Test de evaluare pentru a testa cunoștințele dobândite
- Resurse suplimentare pentru a vă avansa cunoștințele și mai mult

5.2.1 Introducere Minecraft Pi funcții bază

Minecraft Pi este un joc open-world în care jucătorii folosesc blocuri care reprezintă diferite materiale pentru a construi lumi virtuale. Puteți crea orice, de la o singură casă la un castel imens și de la un câmp mic de culturi la un oraș mare.
Minecraft Pi poate fi manipulat folosind scripturi Python care interacționează cu diverse funcții de joc. Ediția Raspberry Pi a Minecraft vine cu o interfață API (Application Programming Interface) care vă permite să controlați jocul folosind programarea Python. Python va fi folosit pentru a manipula blocuri și structuri, pentru a trimite mesaje în joc, pentru a automatiza procesele de construcție și pentru a crea mini-jocuri.
Minecraft Pi acceptă, de asemenea, multiplayer, ceea ce înseamnă că mai mulți jucători pot juca și interacționa între ei pe aceeași hartă. Când mai multe computere STEMKIT sunt conectate prin aceeași rețea Wi-Fi sau Ethernet, modul multiplayer este activat și mai mulți utilizatori se pot conecta la aceeași lume Minecraft.
Pentru a rula Minecraft Pi pe computerul dvs. STEMKIT, trebuie să faceți dublu clic pe pictograma desktop a Minecraft Pi sau să accesați Meniul principal (sigla Raspberry Pi din colțul din stânga sus) →Jocuri **Games → Minecraft Pi** și faceți clic pe el (Figura 1). Când jocul se încarcă, faceți clic pe **Start Game → Create New**.

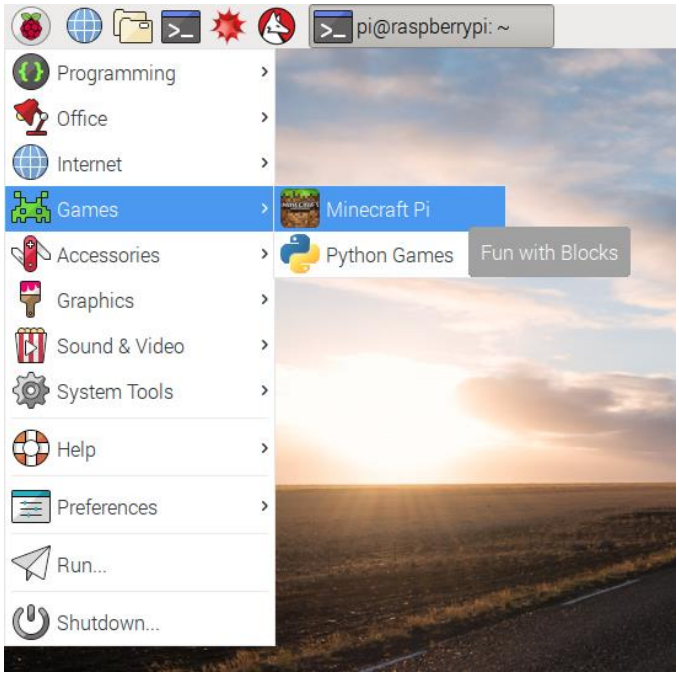


FIGURA 1 RUNNING MINECRAFT PI.

Funcțiile de bază includ navigarea și controlul atunci când intrați într-o lume Minecraft. Folosiți mouse-ul pentru a privi în jur, faceți clic stânga pentru a sparge blocuri și faceți clic dreapta pentru a construi blocuri. Majoritatea funcțiilor de control provin de la tastatură, așa cum se arată în Figura 2.

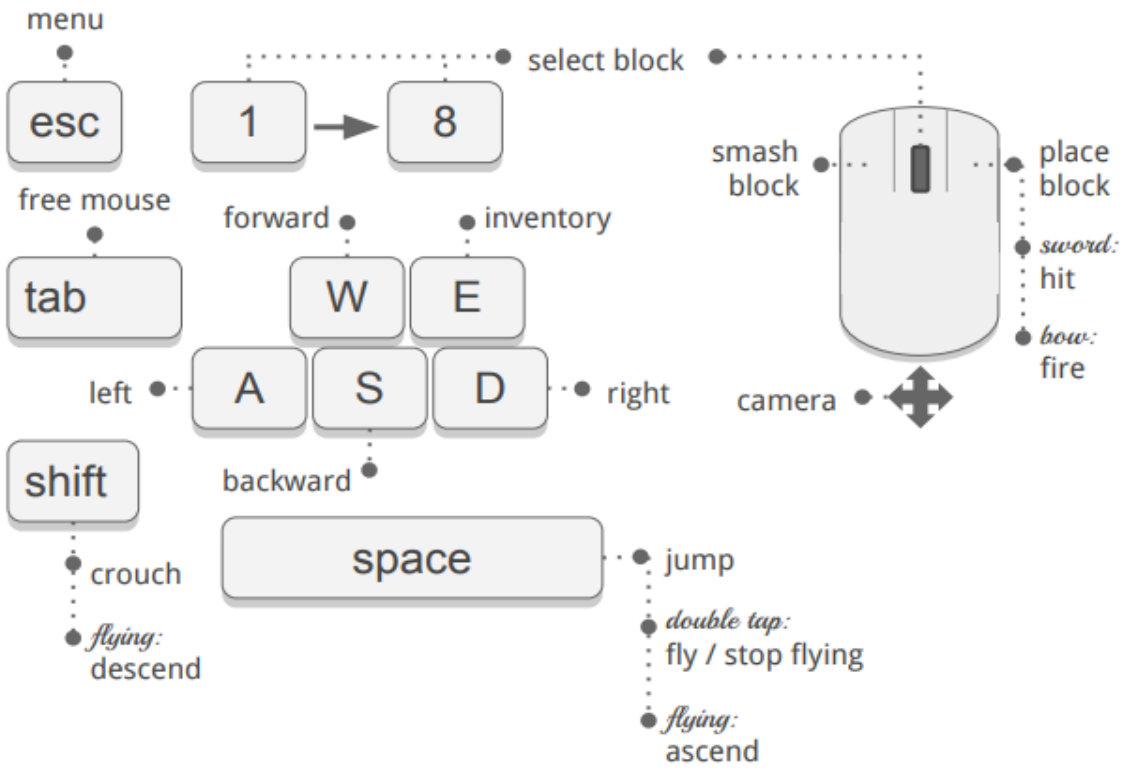


FIGURA 2 MINECRAFT CONTROLS OVERVIEW. SOURCE: HTTPS://ARGHBOX.WORDPRESS.COM/2013/07/28/MINECRAFT-PI-CONTROLS/

5.2.2 Minecraft Pi elemente și jocuri

La prima intrare în joc, utilizatorul începe cu o sabie în mână, care poate fi folosită pentru a elimina blocurile. În partea de jos a ecranului există o vizualizare parțială a inventarului cu diferite tipuri de blocuri și instrumente care pot fi accesate rapid prin rotirea în sus sau în jos a rotii mouse-ului. Aceste articole sunt de obicei cele mai utilizate în joc, dar pot fi schimbate în funcție de ceea ce dorește să construiască utilizatorul și de ce material vrea să aibă acces rapid.

Atingând bara de spațiu, personajul sare. Atingeți de două ori pentru a începe să urcați (zburați în jur). Când bara spațială este eliberată, ascendentul se oprește. Pentru a cădea la pământ, jucătorul trebuie să atingă din nou bara de spațiu din nou.

Apăsând „E”, se va deschide inventarul care conține toate tipurile de blocuri oferite în Minecraft Pi. Răsfoiți și faceți clic stânga pe blocul pe care doriți să îl utilizați.

Apăsarea TAB va elibera mouse-ul din joc, oferind utilizatorului posibilitatea de a utiliza alte programe Raspberry Pi. Când scrieți scripturi, tasta TAB îi va permite utilizatorului să navigheze prin diferite fișiere Python, compilatorul Python sau linia de comandă, în funcție de modul în care utilizatorul dorește să programeze.

Apăsând tasta SHIFT în timpul jocului, personajul se va ghemui. În timpul zborului, tasta SHIFT va face ca personajul tău să coboare.

În cele din urmă, tastele de la 1 la 8 de pe tastatură oferă utilizatorului acces rapid la materiale și instrumente aflate deja în inventarul de acces rapid.

La prima intrare în joc, utilizatorul începe cu o sabie în mână, care poate fi folosită pentru a elimina blocurile. În partea de jos a ecranului există o vizualizare parțială a inventarului cu diferite tipuri de blocuri și instrumente care pot fi accesate rapid prin rotirea în sus sau în jos a rotii mouse-ului. Aceste articole sunt de obicei cele mai utilizate în joc, dar pot fi schimbate în funcție de ceea ce dorește să construiască utilizatorul și de ce material vrea să aibă acces rapid.
Atingând bara de spațiu, personajul sare. Atingeți de două ori pentru a începe să urcați (zburați în jur). Când bara spațială este eliberată, ascendentul se oprește. Pentru a cădea la pământ, jucătorul trebuie să atingă din nou bara de spațiu din nou.
Apăsând „E", se va deschide inventarul care conține toate tipurile de blocuri oferite în Minecraft Pi. Răsfoiți și faceți clic stânga pe blocul pe care doriți să îl utilizați.
Apăsarea TAB va elibera mouse-ul din joc, oferind utilizatorului posibilitatea de a utiliza alte programe Raspberry Pi. Când scrieți scripturi, tasta TAB îi va permite utilizatorului să navigheze prin diferite fișiere Python, compilatorul Python sau linia de comandă, în funcție de modul în care utilizatorul dorește să programeze.
Apăsând tasta SHIFT în timpul jocului, personajul se va ghemui. În timpul zborului, tasta SHIFT va face ca personajul tău să coboare.
În cele din urmă, tastele de la 1 la 8 de pe tastatură oferă utilizatorului acces rapid la materiale și instrumente aflate deja în inventarul de acces rapid: https://www.raspberrypi-spy.co.uk/2014/09/raspberry-pi-minecraft-block-id-number-reference/.

5.2.3 Controlul Minecraft Pi cu Python

Minecraft Pi are un API care permite utilizatorului să comunice și să controleze jocul folosind scripturi Python. API-ul vă permite să scrieți programe care controlează, modifică și interacționează cu lumea Minecraft. Utilizatorul poate crea structuri masive printr-un clic pe buton sau un pod care apare automat permițându-i utilizatorului să meargă peste prăpastii masive sau un joc de mina, un ceas imens în timp real, un tun direcțional programabil sau transforma blocurile în bombe „Hanlon, 2013) (https://www.stuffaboutcode.com/2013/04/minecraft-pi-edition-api-tutorial.html).
Deoarece Minecraft este o lume de cuburi sau blocuri, cu o dimensiune relativă de 1m x 1m x 1m, fiecare bloc are propria sa poziție unică în lume în coordonatele x, y, z (x este înainte / înapoi, z este stânga / dreapta , iar y este sus / jos).
Folosind scripturi Python, puteți face, printre altele, următoarele:

- Obțineți poziția jucătorului.
- Schimbați sau setați poziția jucătorului.
- Obțineți tipul de bloc pe care se află utilizatorul sau se află lângă el
- Schimbați un tip de bloc cu un alt tip de bloc.
- Schimbați unghiurile camerei.
- Postați mesaje către player.

Pentru a deschide interfața Python, accesați Meniul principal Menu → Programming / Programare și faceți clic pe Thonny Python, așa cum se arată în Figura 3.

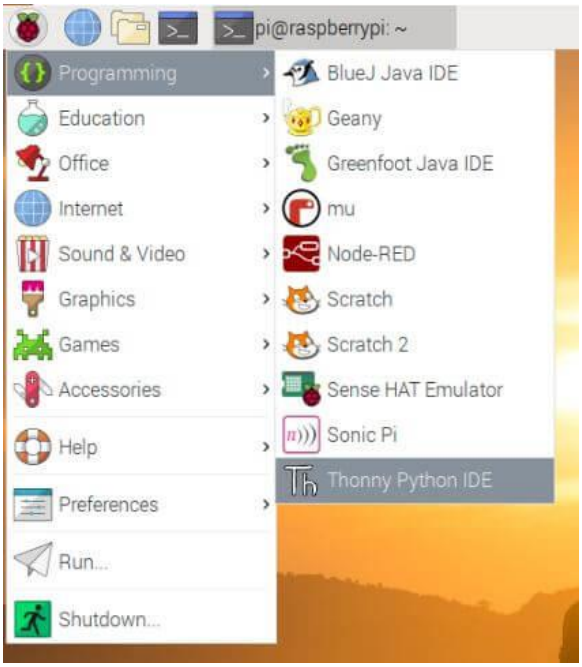


FIGURA 3 LOCALIZARE THONNY PYTHON

În primele noastre linii de cod, vom învăța cum să conectăm Minecraft Pi la scriptul nostru. În primul rând, salvați fișierul folosind numele first.py. Nu uitați să salvați întotdeauna fișierele și să utilizați extensia de fișier .py astfel încât programul dvs. să poată fi interpretat ca fișier Python.
Figura 4 prezintă liniile de cod necesare pentru conectarea Minecraft Pi la Python.

first.py

```
1 from mcpi.minecraft import minecraft
2
3 mc = Minecraft.create()
```

FIGURA 4 CONECTAREA PYTHON LA MINECRAFT PI.

În primul rând, importăm Minecraft. Apoi, salvăm obiectul de joc Minecraft într-o variabilă numită „mc”, care va fi utilizată ulterior pentru a apela comenzi în programul dvs.
Primul nostru program complet va comunica un mesaj personajului nostru din Minecraft. Creați un nou anunț de fișier, salvați-l sub numele hello.py.
Figura 5 arată codul pe care trebuie să îl scrieți:

hello.py

```
1  from mcpi.minecraft import minecraft
2
3  mc = Minecraft.create()
4
5  mc.postToChat("Hello World!")
```

FIGURA 5 TRIMITEREA UNUI MESAJ JUCĂTORULUI.

Când scrieți codul, faceți clic pe Salvare și atingeți F5 pe tastatură pentru a rula scriptul. Apare un mesaj în Minecraft.

Găsirea locației jucătorului:

Găsirea poziției jucătorului este necesară înainte de a construi structuri. Pentru a face acest lucru, utilizăm comanda mc.player.getPos (). Există două moduri de a găsi poziția jucătorului:

1. 1. Folosim o variabilă numită pos și salvăm coordonatele jucătorului, așa cum se arată în Figura 6.

pos.py

```
1  from mcpi.minecraft import minecraft
2
3  mc = Minecraft.create()
4
5  pos = mc.player.getPos()
```

FIGURE 6 SAVING PLAYER'S POSITION ON VARIABLE POS.

2. 2. Salvăm poziția jucătorului în coordonatele xyz, așa cum se arată în Figura 7 (x este înainte / înapoi, z este stânga / dreapta și y este sus / jos)

pos.py *

```
1  from mcpi.minecraft import minecraft
2
3  mc = Minecraft.create()
4
5  x, y, z = mc.player.getPos()
```

FIGURA 7 SALVAREA POZIȚIEI JUCĂTORULUI PE COORDONATELE XYZ.

Teletransportarea jucătorului nostru:

Având coordonatele jucătorului nostru înseamnă, de asemenea, că aceste coordonate le putem manipula, ceea ce înseamnă că ne putem teleporta jucătorul în diferite locuri dintr-o lume Minecraft.

Pentru aceasta folosim comanda mc.player.setPos (). Figura 8 arată cum să folosim această comandă pentru a teleporta playerul nostru cu 200 de spații în aer și apoi să-l urmărim căzând în poziția inițială pe sol.

tel.py

```
1  from mcpi.minecraft import minecraft
2
3  mc = Minecraft.create()
4
5  x, y, z = mc.player.getPos()
6
7  mc.player.setPos(x, y+200, z)
```

FIGURA 8 TELEPORTAȚI JUCĂTORUL NOSTRU 200 DE SPAȚII ÎN AER.

Gestionarea blocurilor:
În afară de găsirea și manipularea poziției jucătorului nostru, putem folosi poziția jucătorului pentru a interacționa cu blocurile. Unul dintre lucrurile pe care le putem face este să cunoaștem blocajul exact pe care se află jucătorul nostru. Pentru a face acest lucru, în primul rând găsim poziția țiglelor folosind comanda mc.getTilePos () și apoi folosim comanda mc.getBlock (x, y, z), așa cum se arată în Figura 9.

tile.py

```
1  from mcpi.minecraft import minecraft
2
3  mc = Minecraft.create()
4
5  x, y, z = mc.getTilePos()
6
7  blockBelowPlayerType = mc.getBlock(x, y, z)
```

FIGURA 9 GĂSIREA POZIȚIEI ȘI SALVAREA ÎN COORDONATE.

Coordinates *xyz* refer to Apoi putem genera blocuri în jurul playerului nostru folosind comanda `mc.setBlock (x, y, z, blockType, blockData)`. Coordonatele xyz se referă la locația în care generăm un bloc, blockType se referă la ID-urile tipului de bloc de diferență, iar block. Data se referă la proprietăți suplimentare pe care unele blocuri le au (de exemplu, culori diferite). Figura 10 arată cum se generează un bloc de piatră (blockType este 1) lângă playerul nostru.

block.py

```
1  from mcpi.minecraft import minecraft
2
3  mc = Minecraft.create()
4
5  x, y, z = mc.player.getPos()
6
7  mc.setBlock(x+1, y, z, 1)
```

FIGURA 10 GENERAREA UNUI BLOC DE PIETRĂ LÂNGĂ JUCĂTOR.

După cum sa menționat, unele blocuri au proprietăți suplimentare. De exemplu:

- Lână (ID-35) □ 0: alb, 1: portocaliu, 2: magenta, 3: albastru deschis, 4: galben etc.
- Lemn (ID-17) □ 0: stejar, 1: molid, 2: mesteacăn etc.
- Iarbă înaltă (ID-31) □ 0: arbust, 1: iarbă, 2: ferigă
- Torță (ID-50) □ 0: îndreptat spre est, 1: vest, 2: nord, 3: sud

Întreaga listă de tipuri de blocuri și proprietăți de blocuri poate fi găsită în următorul link: https://minecraft.gamepedia.com/Block.
În cele din urmă, API-ul permite generarea mai multor blocuri împreună, astfel încât să putem crea diverse structuri la simpla apăsare a unui buton. Pentru aceasta, folosim comanda `setBlocks (x1, y1, z1, x2, y2, z2, blockType, blockData).` Funcționează prin specificarea a două seturi de coordonate pe care API le folosește pentru a umple golul dintre atunci cu un anumit tip de bloc. Figura 11 arată cum se construiește un bloc de 10x10x10 din aur.

goldblock.py

```
1  from mcpi.minecraft import minecraft
2
3  mc = Minecraft.create()
4
5  gold = 41
6
7  x, y, z = mc.player.getPos()
8
9  mc.setBlocks(x+1, y, z+1, x+11, y+11, z+11, gold)
```

FIGURA 11 GENERATING A 10X10X10 BLOCK OF GOLD.

Blocuri speciale:
Blocurile speciale din Minecraft Pi se referă la blocurile care pot interacționa cu împrejurimile unei lumi Minecraft. Aceste blocuri variază de la mecanisme simple la lava care curge. Întreaga listă de blocuri speciale poate fi găsită în următorul link: https://minecraft.gamepedia.com/Block. Există două blocuri speciale care sunt cele mai utilizate în Minecraft: blocurile TNT și Lava.

Blocurile TNT pot fi folosite de jucător pentru a genera explozii controlate care distrug structuri, munți și câmpuri. Jucătorul poate plasa un bloc TNT ca orice alt bloc, dar când faceți clic dreapta pe el de câteva ori, blocul TNT este detonat, oferind jucătorului o fereastră de 4 secunde pentru a se îndepărta înainte de a exploda. ID-ul său de bloc este 46. Figura 12 arată cum se generează blocuri TNT folosind codul Python (rețineți că ultima cifră ar trebui să fie întotdeauna „1”, astfel încât blocul TNT să explodeze.

tnt.py

```
1  from mcpi.minecraft import minecraft
2
3  mc = Minecraft.create()
4
5  tnt = 46
6
7  mc.setBlock(x, y, z, tnt, 1)
```

FIGURA 12 GENERAREA BLOCURILOR TNT FOLOSIND PROGRAMAREA PITONULUI.

Lava este un bloc de lichid care emite lumină care provoacă daune la foc și se răspândește este o zonă de 3x3 deasupra acesteia și o zonă de 5x5 sub ea. Codul său de bloc este 10. Lava poate arde structuri inflamabile, cum ar fi iarba și lemn, dar și jucătorul, așa că trebuie să fim atenți să nu pășim pe el. Când Lava se răcește, devine stâncă. Figura 13 arată cum să generați Lave folosind codul Python.

lava.py

```
1  from mcpi.minecraft import minecraft
2
3  mc = Minecraft.create()
4
5  lava = 10
6
7  mc.setBlock(x+3, y, z+3, lava)
```

FIGURA 13 GENERAREA BLOCURILOR LAVA CU PROGRAMAREA PITONULUI.

Importul modulelor Minecraft Pi:
Pentru a ne asigura că programele noastre funcționează fără probleme, fără a prăbuși jocul, trebuie să importăm câteva module Minecraft la începutul scriptului nostru Python. Aceste module ne permit, de asemenea, să accesăm proprietățile și parametrii necesari pentru a manipula o lume Minecraft. Figura 14 arată unde și cum ar trebui importate aceste module.

modules.py

```
1  from mcpi.minecraft import minecraft
2  from mcpi.block import block
3  from time import sleep
4
5  mc = Minecraft.create()
6
7  # rest of program
8  # ...
9  # ...
10 # ...
```

FIGURA 14 MODULURI NECESARE IMPORTATE ÎN UN SCRIPT PITON.

Recap comenzi Python:

- `postToChat("our message")` – communicate with the player in the game;
- `player.getPos()` – get the precise position of a player;
- • `postToChat („mesajul nostru")` – comunicați cu jucătorul din joc;
- • `player.getPos ()` – obține poziția precisă a unui jucător;
- • `player.setPos (x, y, z)` – setați (modificați) poziția jucătorului;
- • `player.getTilePos ()` – obține poziția blocului în care se află jucătorul în prezent;
- • `getBlock (x, y, z, blockType, blockData)` – obțineți un tip de bloc pentru o anumită poziție;
- • `setBlock (x, y, z, blockType, blockData)` – setați (modificați) un bloc la un anumit tip de bloc;
- • `setBlocks (x1, y1, z1, x2, y2, z2, blockType,` blockData) setează o mulțime de blocuri, toate în același timp, oferind 2 seturi de `coordonate x, y, z.`

5.2.4 Interacțiunea cu lumea fizică prin GPIOGPIO

Această secțiune descrie câteva tutoriale, astfel încât să puteți face Minecraft să interacționeze cu lumea fizică prin GPIO. Veți învăța să programați butoane pentru a interacționa cu jocul Minecraft Pi, pentru a automatiza procesele, pentru a manipula blocuri și pentru a crea mini-jocuri și, în cele din urmă, pentru a programa LED-uri pentru a simula evenimentele care se întâmplă în Minecraft.

Tutorial 1: Conectarea unui buton

În acest tutorial veți învăța cum să conectați un buton prin GPIO al Raspberry Pi. Ar trebui să vă familiarizați cu pinii GPIO, care pinii să utilizați și numerotarea acestora. Figura 15 prezintă numele pinilor standard Broadcom (BCM). Numerotarea nu este în ordine numerică, deci rețineți.

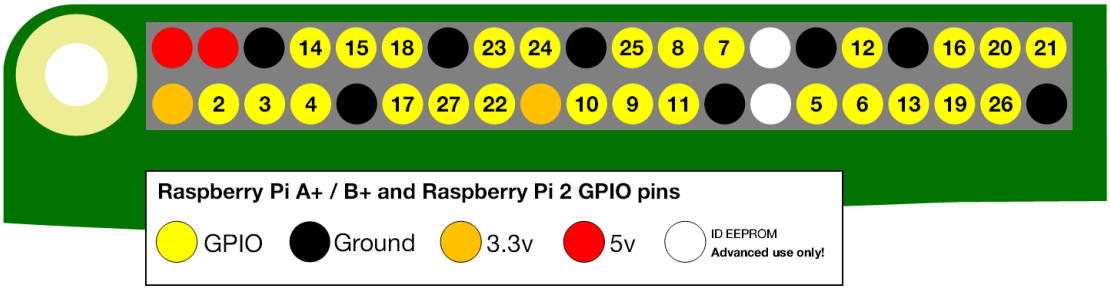


FIGURA 15 GPIO PINS NUMBERING

Materialele cerute in acest tutorial:

- 1 x breadboard /panou de masă
- 1 x push button/ buton
- 1 x 220 Ohm resistor /Rezistor

Conectivitatea este foarte simplă și este prezentată în Figura 16 Rețineți că un cablu jumper este conectat la un pin de 3,3 V (roșu), iar celălalt este conectat la un pin de intrare / ieșire GPIO (numărul pin 16).

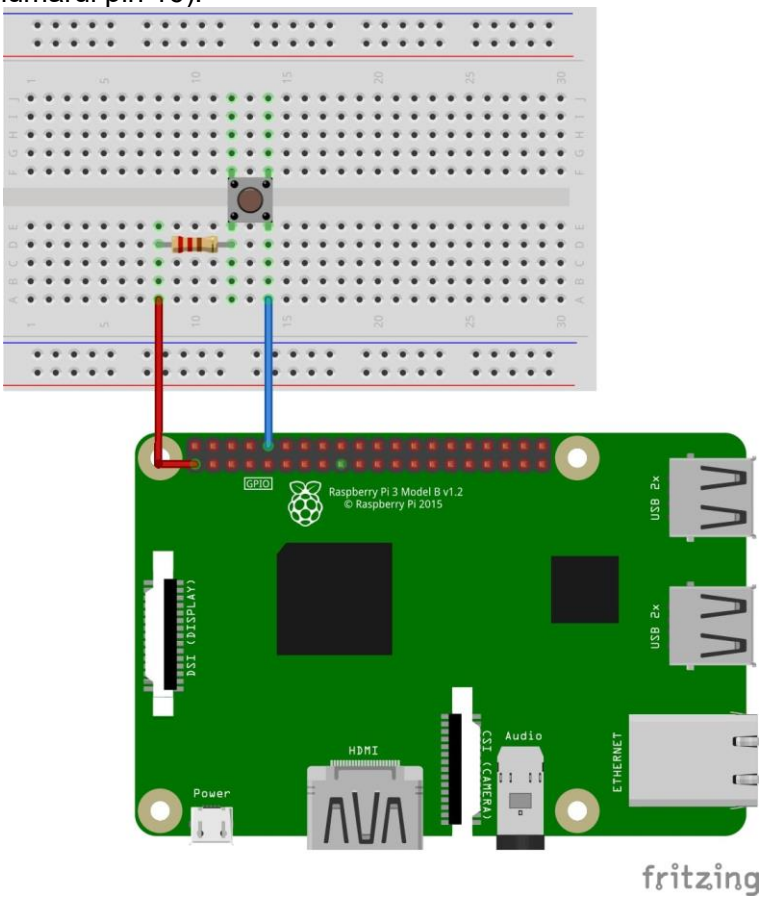


FIGURA 16 CONECTAREA UNUI BUTON PENTRU RASPBERRY GPIO.

SOURCE: RASPBERRYPIHQ.COM

Acum că circuitul dvs. este gata, puteți sări într-un nou fișier Python pentru a testa funcționalitatea butonului. Figura 17 prezintă codul Python necesar, astfel încât să puteți testa butonul.

button_test.py *

```
1  import RPi.GPIO as GPIO
2  import time
3
4  GPIO.setmode(GPIO.BCM) #tell the Pi what headers to use
5  GPIO.setup(15 GPIO.IN) #tell the Pi pin 15 is an input
6
7  while True:
8      if GPIO.input(15) == True: #look for button press
9          print "Button works!" #log result
10         time.sleep(0.5) #wait 0.5 seconds
```

FIGURA 17 PYTHON CODE PENTRU TESTAREA FUNCTIONALITATII BUTONULUI

Salvați și apăsați F5 pentru a rula scriptul. Acum, de fiecare dată când apăsați butonul, mesajul „Butonul funcționează!” ar trebui să apară în fereastra terminalului Python. Pentru a opri scriptul, apăsați *Ctrl+C*.

Tutorial 2: Super mining în Minecraft

În acest al doilea tutorial veți utiliza circuitul anterior pentru a interacționa cu jocul dvs. Minecraft Pi. Veți crea un program care distruge un bloc de blocuri într-o lume Minecraft de fiecare dată când apăsați butonul. Creați un fișier nou și salvați-l ca mining.py.
Urmați codul din Figura 18 și vedeți ce se întâmplă în lumea Minecraft când rulați scriptul (F5) și apăsați butonul. Rețineți că puteți manipula tipurile de blocuri și coordonatele după cum doriți, dar nu înnebuniți prea mult, deoarece Raspberry Pi s-ar putea lupta.In this second tutorial you will use the previous circuit to interact with your Minecraft Pi game.

mining.py

```
import RPi.GPIO as GPIO
import sleep
from mcpi.minecraft import Minecraft

mc = Minecraft.create() #connect Minecraft Pi with Python

GPIO.setmode(GPIO.BCM) #tell the Pi what headers to use
GPIO.setup(15, GPIO.IN) #tell the Pi pin 15 is an input

while True:
    if GPIO.input(15) == True #look for button press
    x, y, z = mc.player.getPos() #read the player's position

    mc.setBlocks(x, y, z, x+10, y+10, z+10, 0) #mine 10 blocks
    mc.setBlocks(x, y, z, x-10, y-10, z-10, 0) #mine 10 blocks

    time.sleep(0.5) #wait 0.5 seconds
```

FIGURA 18 COD PITON PENTRU TESTAREA FUNCȚIONALITĂȚII BUTONULUI

Tutorial 3: Crearea detectorului diamant

În al treilea tutorial veți crea un program care detectează blocuri de diamante în Minecraft. Mai întâi trebuie să creați un nou circuit folosind următorul material:

- 1 x panou de măsurare
- 1 x LED (orice culoare)
- Rezistor 1 x 220 Ohm
- 2 x cabluri jumper female-to-male / cables de la mamă la mamă

Conectivitatea este foarte simplă și este prezentată în Figura 19. Rețineți că un cablu jumper este conectat la un pin de masă (negru), iar celălalt este conectat la un pin de intrare / ieșire GPIO (numărul pin 23).

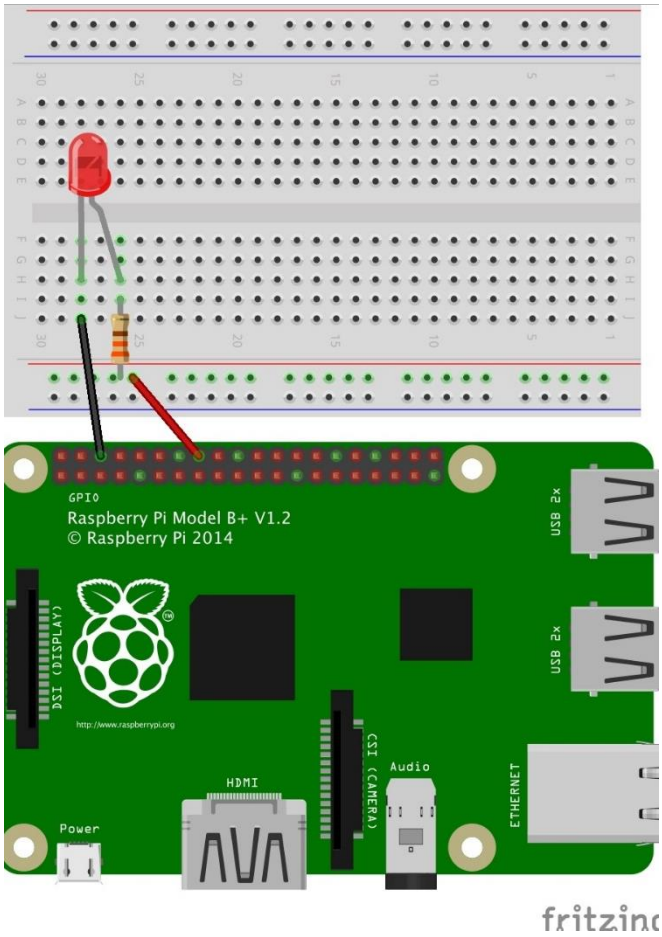


FIGURA19 CONECTAREA UNUI LED LA GPIO.
SOURCE: RASPBERRYPIHQ.COM

După ce ați creat circuitul, puteți deschide Python, puteți crea un fișier nou și îl puteți salva ca detector.py. Urmați codul din Figura 20, rulați scriptul (F5) și deplasați-vă în lumea Minecraft.When you have created the circuit, you can open Python, create a New File and Save it as *detector.py*.

detector.py

```python
import RPi.GPIO as GPIO
import time
from mcpi.minecraft import Minecraft
from mcpi.blocks import Blocks

mc = Minecraft.create() #connect Minecraft with Python

led_pin = 23 #store the GPIO pin number in variable

GPIO.setmode(GPIO.BCM) #tell the Pi what headers to use
GPIO.setup(23, GPIO.OUT) #tell the Pi pin 23 is an output

while True: #repeat indefinitely
    x, y, z = mc.playerPos()
    for i in range(10): #check every block until block 10
        if mc.getBlock(x, y-i, z) == 56:
            GPIO.output(led_pin, True) #turn LED on
            time.sleep(0.5) #wait
            GPIO.output(led_pin, False) #turn LED off
            time.sleep(0.5) #wait
```

FIGURA 20 COD PITON PENTRU CREAREA UNUI DETECTOR DE DIAMANTE

5.2.5 Importul de hărți noi și pachete de resurse

Minecraft Pi oferă posibilitatea de a importa noi hărți și pachete de resurse pentru a explora lumi noi și a vă îmbunătăți experiența de joc. Adăugarea de hărți noi este o procedură ușoară:

1. Mai întâi trebuie să găsiți o lume care să vă placă și să o descărcați pe Raspberry.

2. Când găsiți unul care vă place, trebuie să îl descărcați ca fișier .zip. Pentru a face acest lucru, faceți clic pe descărcare și urmați pașii.

3. Când fișierul este descărcat, trebuie să extrageți fișierul ***.zip***. Faceți clic dreapta și apoi faceți clic pe extragere.

4. În sistemul de operare Raspberry Pi, deschideți sistemul de fișiere ***filesystem***. Selectați vizualizare ***view*** și apoi afișați fișierele ascunse ***show hidden files***.

5. Găsiți folderul Minecraft, faceți dublu clic pe el, apoi faceți dublu clic din nou pe folderul numit „***jocuri***” și apoi faceți dublu clic din nou pe folderul numit „***com.mojang***”. Acolo veți găsi un folder numit „***minecraftWorlds***”. Faceți dublu clic și sunteți gata să adăugați lumi noi. Pur și simplu trageți și fixați lumile dezarhivate în dosar.

6. ***Sfat bonus:*** în timp ce vă aflați în directorul lumilor Minecraft, puteți ***redenumi*** lumile pe care le-ați creat deja în joc, făcând clic dreapta pe ele și apoi redenumiți. Acesta este singurul mod de a vă redenumi lumile.

Veți găsi o mare varietate de hărți în următoarele link-uri:

- https://thebraithwaites.co.uk/minecraft-pi-edition-maps-texture-packs-survival-and-more/
- https://www.minecraftforum.net/forums/minecraft-pocket-edition/mcpe-maps?page=2

Când vine vorba de pachete de resurse, procedura este similară cu cea de mai sus, dar este necesar un alt director.
De exemplu, puteți descărca un pachet de texturi de aici. Urmați instrucțiunile de pe link. Trebuie să găsiți folderul „***Assets***" în directorul Minecraft și să plasați acolo fișierele dezarhivate. Încărcați din nou jocul și pachetul de texturi ar trebui să fie gata de utilizare.
Mai multe pachete de resurse împreună cu hărți pot fi găsite în următoarele link-uri:

- https://www.planetminecraft.com/collection/10099/resource-packs/
- https://www.planetminecraft.com/texture_pack/the-pi-pack---132---037/
- https://thebraithwaites.co.uk/minecraft-pi-edition-maps-texture-packs-survival-and-more/
- https://www.minecraftforum.net/forums/minecraft-pocket-edition/mcpe-maps?page=2

5.3 Example Practice

Pentru a vă testa cunoștințele și a practica mai departe, vă propunem câteva exemple practice. Acestea se recomandă a fi realizate după fiecare tutorial și includ următoarele:

- Exemplul 1: Trap player între blocuri
- Exemplul 2: Construiește o casă
- Exemplul 3: Construiți o casă cu o întorsătură
- Exemplul 4: bloc mare de blocuri
- Exemplul 5: Creați un vulcan (avansat)

5.3.1 Exemplul 1: Capcană jucător între blocuri!

Codul de mai jos obține poziția țiglelor jucătorilor, apoi apelează funcția getBlock () a API-ului Minecraft pentru a afla tipul de bloc pe care stă jucătorul (scăzând 1 din coordonata y) înainte de a utiliza `setBlock` () pentru a crea blocuri de același tip pe care se află jucătorul în jurul său (https://www.stuffaboutcode.com/2013/04/minecraft-pi-edition-api-tutorial.html).
De exemplu, dacă jucătorul tău stă pe STONE, atunci blocurile STONE vor apărea în jurul lui.

trap.py *

```
from mcpi.minecraft import minecraft
from mcpi.block import block
from time import sleep

mc = Minecraft.create()

playerTilePos = mc.player.getTilePos()
blockBelowPlayerType = mc.getBlock(playerTilePos.x, playerTilePos.y - 1, playerTilePos.z)
mc.setBlock(playerTilePos.x + 1, playerTilePos.y + 1, playerTilePos.z, blockBelowPlayerType)
mc.setBlock(playerTilePos.x, playerTilePos.y + 1, playerTilePos.z + 1, blockBelowPlayerType)
mc.setBlock(playerTilePos.x - 1, playerTilePos.y + 1, playerTilePos.z, blockBelowPlayerType)
mc.setBlock(playerTilePos.x, playerTilePos.y + 1, playerTilePos.z - 1, blockBelowPlayerType)
mc.postToChat("Trapped you")
time.sleep(5)
```

Utilizați codul de mai sus, împreună cu ceea ce ați învățat până acum și scrieți scriptul. Apoi rulați scriptul în interfața de programare Python în timp ce rulați Minecraft și vedeți ce se întâmplă. Este jucătorul tău blocat între blocul de același tip cu cel care stă? Dacă nu, ce este în neregulă? Poți găsi greșeala? Dacă totul funcționează conform descrierilor, atunci următorul lucru de făcut este să vă eliberați playerul. Încearcă să faci asta singur!
* Sugestie: eliminarea blocurilor se face folosind `setBlock` (), dar mai degrabă decât să facem blocul solid ca *WOOD sau STONE*, l-am setat la AIR.

5.3.2 Examplul 2: Construiește o casă!

Vrei să construiești o casă, dar nu vrei să petreci ore întregi construind blocuri unul câte unul. Următorul cod va crea de la zero o clădire simplă cu aspect de casă. Codul este destul de simplu și explicat mai jos. Diferite linii de comentarii sunt inserate pentru comoditate.

house.py *

```
1  from mcpi.minecraft import minecraft
2  from mcpi.block import block
3  from time import sleep
4
5  mc = Minecraft.create()
6
7  x, y, z = mc.player.getPos()
8
9  # build walls
10 mc.setBlocks(x+2, y-1, z+2, x+7, y+3, z+8, 5)
11 # remove interior
12 mc.setBlocks(x+3, y, z+3, x+6, y+2, z+7, 0)
13 # make doorway
14 mc.setBlocks(x+2, y, z+5, x+2, y+1, z+5, 0)
15 # make window 1
16 mc.setBlocks(x+4, y+1, z+8, x+5, y+1, z+8, 102)
17 # make window 2
18 mc.setBlocks(x+4, y+1, z+2, x+5, y+1, z+2, 102)
```

Destul de simplu, nu? Utilizați codul de mai sus, împreună cu ceea ce ați învățat până acum și scrieți scriptul. Apoi rulați scriptul în interfața de programare Python în timp ce rulați Minecraft și vedeți ce se întâmplă. Ce zici de adăugarea unui al doilea etaj și a unui acoperiș? Folosiți-vă imaginația și abilitățile de programare împreună cu comanda mc.setBlocks () pentru a modifica codul și a vedea ce se întâmplă!

5.3.3 Examplul 3: Construiește o casă cu o întorsătură

Ați învățat să construiți o casă rulând scriptul, dar ce zici de construirea oricâtor case doriți și oriunde doriți, cu doar clicul butonului?
Puteți atribui un buton de jocuri arcade pentru a construi case de fiecare dată când loviți. Cum?
În primul rând, trebuie să conectați un buton la placa GPIO așa cum ați învățat (asigurați-vă că vă conectați la pinii potriviți de pe placa GPIO. Acest exemplu *folosește GPIO24 ca pin de intrare, dar puteți utiliza ceea ce se potrivește cel mai bine nevoilor dvs. și puteți modifica cod în consecință).*
Apoi, urmați scriptul de mai jos (pentru mai multă comoditate sunt inserate diverse linii de comentarii):

house_twist.py *

```
#connect Python with Raspberry Pi GPIO board
import RPi.GPIO as GPIO

from mcpi.minecraft import minecraft
from mcpi.block import block
from time import sleep

mc = Minecraft.create()

GPIO.setmode(GPIO.BCM) #set mode for GPIO
GPIO.setup(24, GPIO.IN) #set input pin to GPIO24

while True:
    if GPIO.input(24) == True:
       x,y,z = mc.player.getPos() #get players position
       mc.setBlocks(x+2,y-1,z+2,x+7,y+3,z+8, 5) #make shell
       mc.setBlocks(x+3,y,z+3,x+6,y+2,z+7, 0) #remove inside
       mc.setBlocks(x+2,y,z+5,x+2,y+1,z+5, 0) #make doorway
       mc.setBlocks(x+4,y+1,z+8,x+5,y+1,z+8, 102) #make window 1
       mc.setBlocks(x+4,y+1,z+2,x+5,y+1,z+2, 102) #make window 2
       mc.setBlocks(x+7,y+1,z+4,x+7,y+1,z+6, 102) #make window 3
       print ("House is built")
       time.sleep(0.1) #wait 0.1 sec
```

Rulați scriptul și apoi apăsați butonul atribuit. O casă ar trebui să apară lângă tine. Vrei să construiești mai mult? Apăsați butonul de mai multe ori în timp ce vă deplasați în lumea Minecraft. Folosiți-vă imaginația și abilitățile de programare și editați coordonatele `mc.setBlocks ()` pentru a adăuga mai multe etaje, pentru a extinde aria structurii, pentru a crea mai multe uși și ferestre etc. Puteți programa un buton pentru a construi un bloc de clădiri?

5.3.4 Exemplul 4: Big Block of blocks

Un alt exercițiu folosind comanda mc.setBlocks (). Puteți crea blocuri masive de un anumit tip de bloc pentru utilizare ulterioară. Încercați să utilizați următorul cod în script și să vedeți ce se întâmplă când rulați programul.

```
tnt = 46
mc.setBlocks (x + 1, y + 1, z + 1, x + 11, y + 11, z + 11, 46, 1)
```

Utilizați codul de mai sus, împreună cu ceea ce ați învățat până acum și scrieți scriptul. Apoi rulați scriptul în interfața de programare Python în timp ce rulați Minecraft și vedeți ce se întâmplă. Funcționează? Dacă nu, ce este în neregulă? Dacă totul funcționează în conformitate cu descrierea, atunci următorul lucru de făcut este să schimbați tipurile și formele de blocuri și să vă construiți structurile unice. Încearcă să faci asta singur!

5.3.5 Exemplul 5: Creati un Vulcan! (avansat)

Următorul cod vă va arăta cum să creați un vulcan activ în lumea Minecraft de la zero. Deoarece acest exercițiu este avansat comparativ cu cel anterior, întregul cod este dat mai jos (https://learnlearn.uk/raspberrypi/2018/02/10/minecraft-python-volcano-tutorial/). Diferite linii de comentarii sunt inserate pentru clarificare.

volcano.py

```
# import Minecraft
import mcpi.minecraft as Minecraft
# import block library
import mcpi.block as block
# import random and time modules
import random, time

mc = Minecraft.create()
mc.postToChat("Minecraft Volcano!")

mc.setBlocks(-100, 0, -100, 100, 50, 100, block.AIR)

height = 20
center = 20,0,0  #x,y,z

for i in range(height):     # Create the base!
    size = height - i
    mc.setBlocks(center[0] - size, center[1] + i, center[2] - size, center[0] + size, center[1] + i, center[2] + size, block.STONE)

for i in range(height * height):
    randomx = random.randint(center[0] - height, center[0] + height)
    randomz = random.randint(center[2] - height, center[2] + height)
    mc.setBlock(randomx,center[1] + height + 10, randomz, block.GRAVEL)

while True:
    mc.setBlock(center[0],center[1]+height, center[2], block.LAVA_FLOWING)
    time.sleep(1)
```

Utilizați codul de mai sus, împreună cu ceea ce ați învățat până acum și scrieți scriptul. Apoi rulați scriptul în interfața de programare Python în timp ce rulați Minecraft și vedeți ce se întâmplă. Funcționează? Dacă nu, ce este în neregulă? Dacă totul funcționează conform descrierii, atunci următorul lucru de făcut este să-l faci pe vulcan să erupe apă în loc de lavă. Dacă a fost prea ușor, încercați să creați un vulcan circular în loc de unul pătrat.

5.4 Test de evaluare

1. Minecraft Pi acceptă caracteristicile complete ale jocului Minecraft.
    1. Da
    2. **Nu**
2. Pot rula Minecraft prin:
    1. Faceți dublu clic pe pictograma desktop
    2. Navigarea la meniul principal
    3. Folosind linia de comandă pentru a rula programul
    4. **Toate cele de mai sus**
3. Minecraft Pi are un API pentru a controla jocul folosind interfața de programare Python și un număr de scripturi.
    1. **Da**
    2. Nu
4. Un bloc in Minecraft are $1m^3$.
    1. Da
    2. Nu
    3. **Nu tot timpul**
5. Ce face următoarea linie de cod: *din mcpi.minecraft import Minecraft:*
    1. Codul este greșit

2. **Conectează interfața de programare Python la Minecraft**
3. Schimbă unghiul camerei în joc
4. Niciuna dintre cele de mai sus

6. Ne putem teleporta playerul folosind comanda setPos ().
   1. **Da**
   2. Nu
7. Ce înseamnă următoarea linie de cod, aur = 41
   1. Am stabilit prețul unui articol la 41 de unități de aur
   2. Am setat 41 de blocuri de aur pentru a fi utilizate în joc
   3. **41 reprezintă ID-ul blocului aurului salvat într-o variabilă numită aur**
   4. Niciuna dintre cele de mai sus
8. Ce înseamnă ultima cifră în următoarea linie de cod, mc.setBlock(x, y, z, wool, **2**):
   1. **Proprietate suplimentară - block type wool**
   2. Solicităm 2 blocuri de lână/wool
   3. Reprezintă un fel de coordonate
   4. Niciuna dintre cele de mai sus
9. Linia *GPIO.setup(23, GPIO.IN)* îi spune Pi că acest pin 23 este folosit ca intrare/ input.
   1. **Da**
   2. Nu
10. Pentru a crea un circuit pentru utilizarea unui buton cu Raspberry Pi, am nevoie de un panou de calcul, două cabluri jumper și buton.
    1. Da
    2. **Nu**

5.5 Referinte

- Richardson, C., (2013), Minecraft Pi book, retrieved from https://arghbox.files.wordpress.com/2013/06/minecraftbook.pdf
- Minecraft controls, retrieved from https://arghbox.wordpress.com/2013/07/28/minecraft-pi-controls/
- Block ID numbers, retrieved from https://www.raspberrypi-spy.co.uk/2014/09/raspberry-pi-minecraft-block-id-number-reference/
- O'Hanlon, (2013), Minecraft: Pi Edition - API Tutoria, retrieved from https://www.stuffaboutcode.com/2013/04/minecraft-pi-edition-api-tutorial.html
- Special blocks in Minecraft, retrieved from https://minecraft.gamepedia.com/Block
- Minecraft Wiki, retrieved from https://minecraft.gamepedia.com/Pi_Edition

5.6 Resurse aditionale

Dacă doriți să accesați mai multe detalii despre Minecraft Pi, vă recomandăm să vizitați următoarele resurse:

- Minecraft API: https://www.stuffaboutcode.com/p/minecraft-api-reference.html
- Raspberry Pi: https://www.stuffaboutcode.com/p/raspberry-pi.html
- Minecraft Wiki: https://minecraft.gamepedia.com/Pi_Edition

- Manhattan project in Minecraft: https://www.stuffaboutcode.com/2013/04/minecraft-pi-edition-manhattan-stroll.html
- Massive Analogue Clock: https://www.stuffaboutcode.com/2013/02/raspberry-pi-minecraft-analogue-clock.html
- Planetary gravity simulation: https://www.stuffaboutcode.com/2013/03/raspberry-pi-minecraft-planetary.html
- Coding Tips: http://www.laschina.org/wp-content/uploads/2017/09/Minecraft-Coding-Tips.pdf
- Raspberry Pi projects: https://projects.raspberrypi.org/en/projects?software%5B%5D=python&hardware%5B%5D=raspberry-pi
- Minecraft Pi Handbook: http://repository.erasmusplus.website/RETROSTEM/Material/Minecraft%20Pi%20-%20Handbook.pdf

5.7 Concluzii

Puncte cheie de concluzie pentru **Modulul 3 - Introducere în Minecraft Pi**:
Dacă ați urmărit această resursă cu Raspberry Pi, vă așteptați să:

- Accesați Minecraft Pi și creați o lume nouă.
- Navigați în jurul Minecraft Pi utilizând comenzile de mișcare de pe tastatură.
- Știți cum să plasați și să distrugeți un bloc și să navigați prin diferite tipuri de blocuri din inventarul din joc.
- Conectați Python la Minecraft Pi.
- Utilizați interfața de programare Python.
- Manipulați blocuri folosind cod și scripturi Python.
- Înțelegeți bine restul funcțiilor Minecraft Pi.
- Faceți Minecraft să interacționeze cu lumea exterioară folosind butoane și LED-uri.
- Importați lumi noi și descoperiți pachete de resurse compatibile cu Minecraft Pi.Connect Python to Minecraft Pi.

6 Raspberry Pi GPIO Programare folosind Python [P2-AKNOW]

6.1 Glosar de termeni

Term / Concept	Definitie / Explacatii
Raspberry Pi GPIO	Raspberry Pi GPIO este rândul de pini de-a lungul marginii superioare a plăcii. Un antet cu 40 de pini se găsește pe toate plăcile Raspberry Pi actuale. Cea mai mare parte a funcționalității Raspberry Pi provine de la acești pini care pot fi configurați și controlați utilizând un limbaj de programare. Oricare dintre pinii GPIO poate fi desemnat în software ca un pin de intrare sau de ieșire și utilizat pentru o gamă largă de scopuri, cum ar fi controlul LED-urilor, buzzerelor, motoarelor, servo-urilor, interacțiunii cu senzorii, comunicarea cu alte dispozitive etc.
Python	Python este un limbaj de programare interpretat, orientat spre obiecte, la nivel înalt. Python are o sintaxă simplă, ușor de învățat, care subliniază lizibilitatea și, prin urmare, reduce timpul total necesar pentru ao învăța și pentru a dezvolta și menține un program. Python acceptă module și pachete, ceea ce încurajează modularitatea programului și reutilizarea codului. Interpretul Python și biblioteca standard extinsă sunt disponibile sub formă sursă sau binară fără taxe pentru toate platformele majore și pot fi distribuite în mod liber.

6.2 Continut

Modul 4 – Raspberry Pi GPIO programarea utilizând Python oferă o introducere a utilizării pinilor de ieșire de intrare generală (GPIO) de pe computerul dvs. Raspberry Pi. GPIO este o caracteristică puternică a Raspberry Pi și poate fi găsit pe marginea superioară a plăcii. Un antet GPIO cu 40 de pini vine cu toate computerele Raspberry Pi actuale și cea mai mare parte a funcționalității sale provine de la acești pini. După finalizarea modulului 4, se așteaptă ca elevul să știe cum să controleze și să interacționeze cu Raspberry Pi GPIO folosind limbajul de programare Python.
Modulul 4 constă din următoarele elemente:

- Introducere în Raspberry Pi GPIO
- Introducere în limbajul de programare Python
- Introducere în circuitele electronice
    - Clipește un LED folosind pinii Raspberry Pi GPIO și Python
    - Activați un buzzer cu un buton
    - Conectați și utilizați un senzor prin GPIO
    - Creați o alarmă de proximitate sau o lumină de stop
- Exemple practice de utilizare a GPIO cu programare Python
- Test de evaluare pentru a testa cunoștințele dobândite

- Resurse suplimentare pentru a vă avansa cunoștințele și mai mult

6.2.1 Introducere in Raspberry Pi GPIO pins

Raspberry Pi GPIO este rândul de pini de-a lungul marginii superioare a plăcii. Un antet cu 40 de pini se găsește pe toate modelele Raspberry Pi actuale. Majoritatea funcționalităților Raspberry Pi provin de la GPIO, care poate fi configurat și controlat folosind un limbaj de programare. Figura 1 arată unde se află pinii GPIO pe un Raspberry Pi.

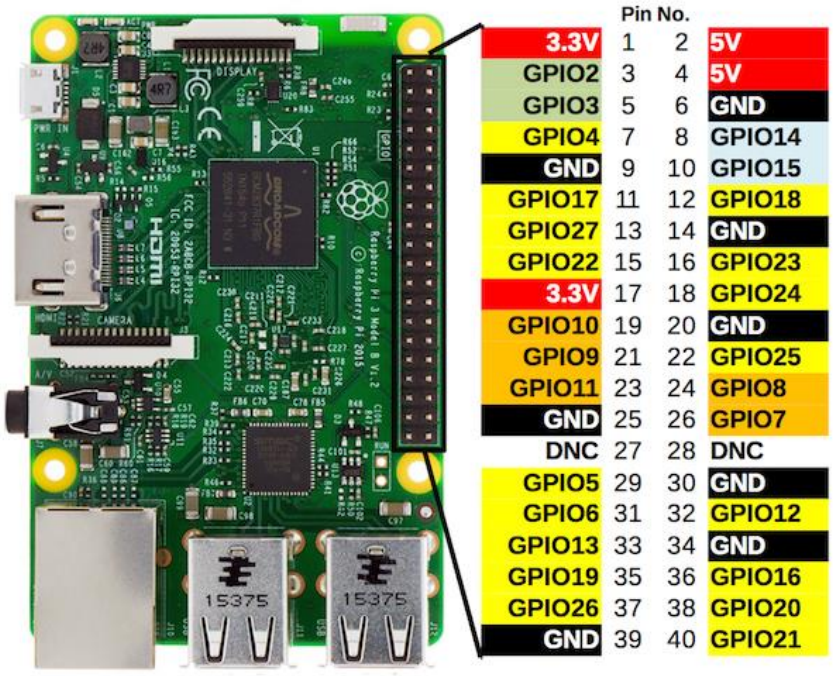


FIGURA 21 RASPBERRY PI GPIO PINS

Oricare dintre pinii GPIO poate fi desemnat în software ca un pin de intrare sau de ieșire și utilizat pentru o gamă largă de scopuri, cum ar fi controlul LED-urilor, buzzerelor, motoarelor, servo-urilor, interacțiunii cu senzorii, comunicarea cu alte dispozitive etc.
Utilizatorul trebuie să se familiarizeze cu pinii GPIO, care dintre ace pot fi folosiți, denumirea și numerotarea acestora și ce proprietăți au. Figura 2 prezintă numele pinilor standard Broadcom (BCM). Rețineți că numerotarea pinilor GPIO nu este în ordine numerică.

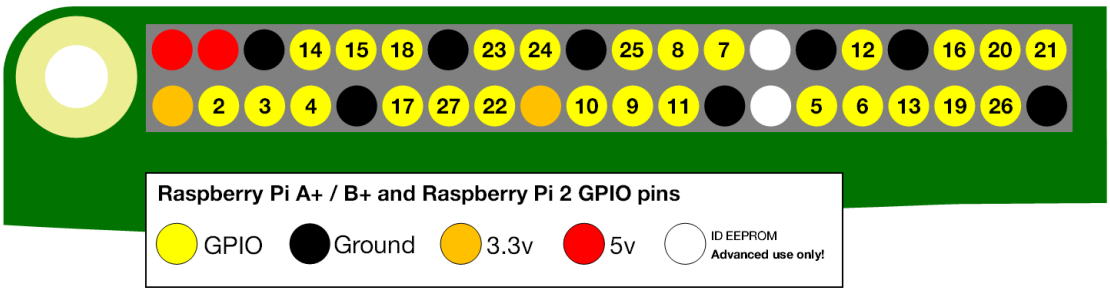


FIGURA 22 STANDARD BROADCOM (BCM) PIN NUME

Pentru a le compara cu Raspberry Pi, orientați pinii astfel încât să fie în partea stângă sus a plăcii, așa cum se arată în Figura 3.

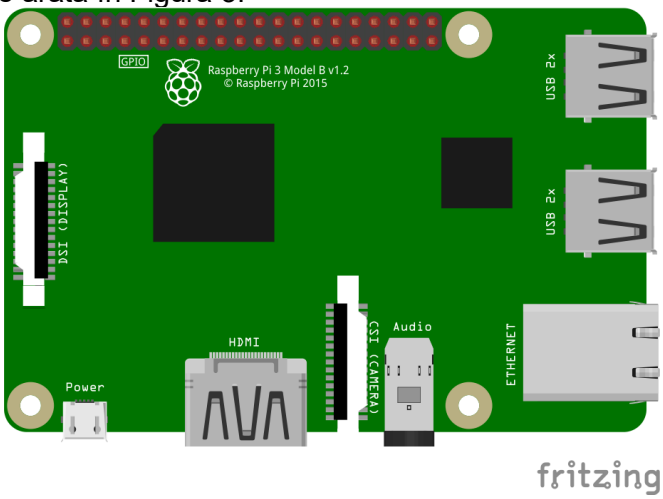


FIGURA 23 SCHEMATIC TOP VIEW OF THE RASPBERRY PI COMPUTER

Pinii de tensiune și împământare /Voltage and Ground pins: doi pini de 5V și doi pini de 3,3V sunt prezenți pe placă, precum și un număr de pini de masă (0V), care nu pot fi configurate. Pinii de masă ar trebui considerați ca fiind unul dintre cei mai importanți pin, deoarece orice circuit creat ar trebui să fie conectat la unul dintre ei, altfel ar putea deteriora placa sau poate avea un comportament neașteptat. Pinii rămași sunt pini de 3,3V de uz general, ceea ce înseamnă că ieșirile sunt setate la 3,3V și intrările sunt de așteptat să fie de 3,3V.
Outputs /Ieșiri: un pin GPIO desemnat ca pin de ieșire poate fi setat la ridicat (3,3V) sau scăzut (0V).
Inputs /Intrări: un pin GPIO desemnat ca pin de intrare poate fi citit ca fiind ridicat (3,3V) sau scăzut (0V).
Pinii GPIO au practic două stări: HIGH sau LOW și, combinând aceste opțiuni binare, se pot crea mult mai multe rezultate cu circuite pentru a interacționa cu lumea fizică cu programe. În următorul capitol vor fi utilizate semnale înalte și joase pentru a face Raspberry Pi să interacționeze cu componentele electronice și senzorii.
Este important să distingem pinii GPIO. Unele persoane folosesc etichete cu pini, de exemplu imprimabile Raspberry Leaf (Figura 4).

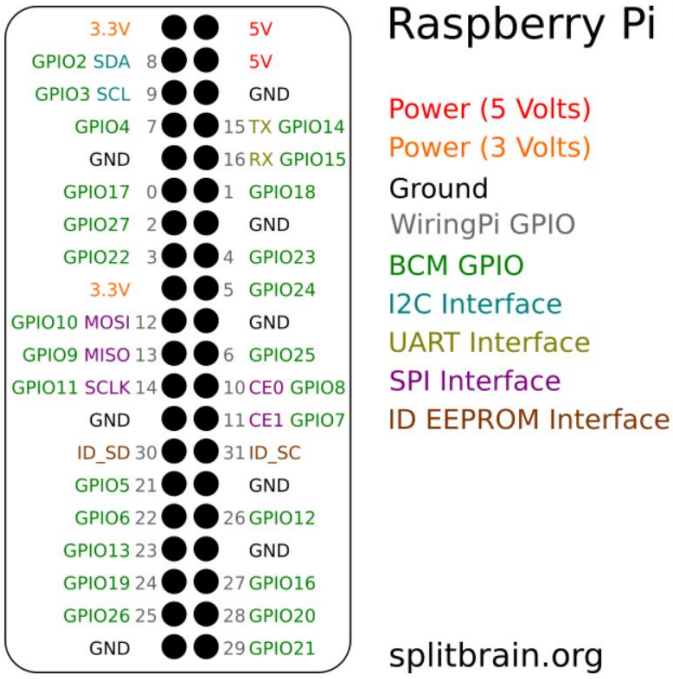


FIGURA 24 RASPBERRY PI B+ LEAF

Mai mult, o referință la îndemână poate fi accesată pe Raspberry Pi deschizând o fereastră de terminal și executând comanda pinout (Figura 5). Acest instrument este preinstalat pe imaginea desktop Raspberry Pi OS și este furnizat de biblioteca GPIO Zero Python.

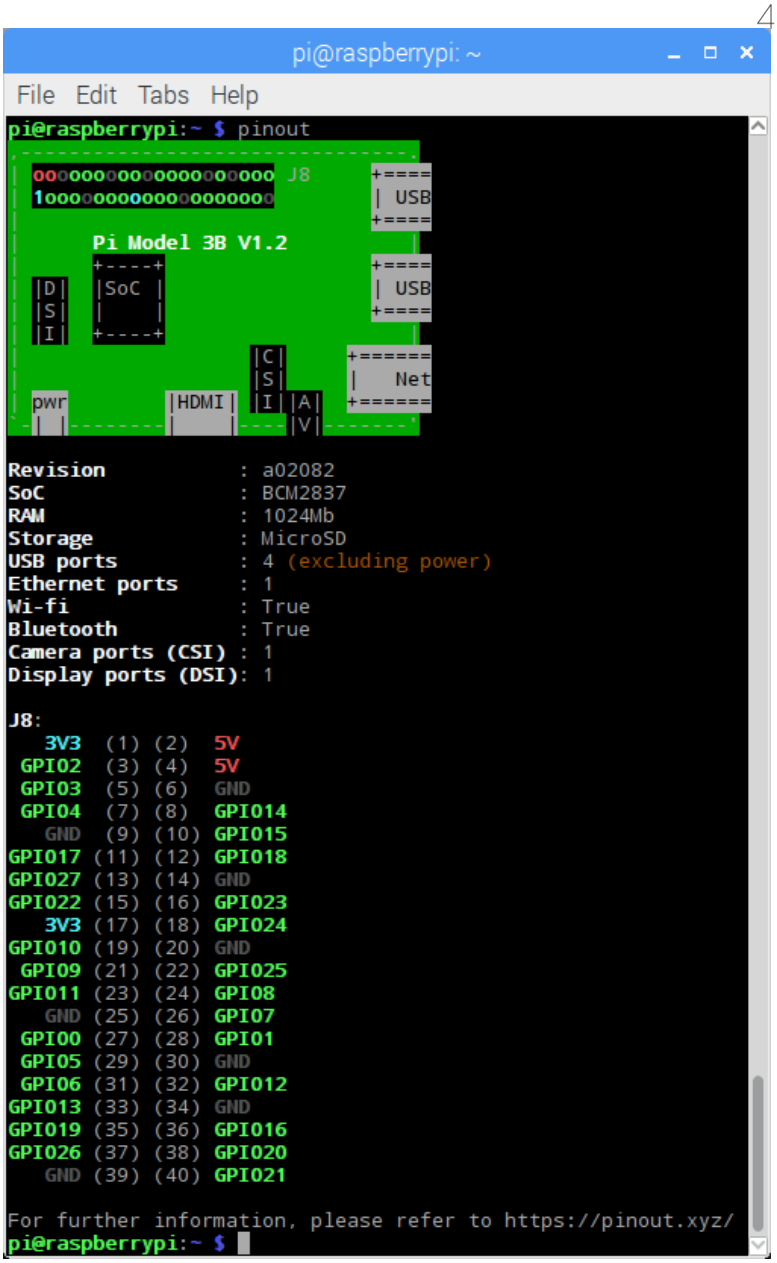


FIGURA: 25 RASPBERRY PI PINOUT TOOL

Este posibil să controlați pinii GPIO folosind un număr de limbaje și instrumente de programare. În modulul 4, veți învăța să utilizați limbajul de programare Python pentru a interacționa cu GPIO-ul Raspberry Pi.

6.2.2 Introducere in Python limbaj de programare

Python este un limbaj de programare orientat pe obiecte de uz general, ușor de învățat, citit și scris, iar cu Raspberry Pi, vă permite să vă conectați programul la lumea fizică.Python are o sintaxă foarte curată, ușor de citit și care folosește cuvinte cheie

standard în limba engleză.

FIGURA 26 PYTHON LOGO

Raspberry Pi are un mediu de dezvoltare Python 3 preinstalat, numit Thonny. Puteți deschide Thonny din meniul de aplicații de pe desktopul dvs. Raspberry Pi. Thonny este echipat cu un REPL (Read-Evaluate-Print-Loop) care este un prompt în care puteți introduce comanda Python. Thonny are o sintaxă de evidențiere încorporată și un anumit suport pentru completarea automată, care vă ușurează viața atunci când introduceți cod.

Python acceptă următoarele:

- Print
- Indentare
- Variabile
- Comentarii
- Liste
- Iterații
- Gamă
- Lungime
- Dacă afirmațiile •

```
print("Hello world")
```

FIGURA 27 SIMPLE EXAMPLE OF PYTHON SYNTAX

Pentru informații detaliate despre modul în care este utilizat fiecare dintre cele de mai sus, puteți găsi urmând acestea links

1. https://www.raspberrypi.org/documentation/usage/python/README.md
2. https://www.raspberrypi.org/documentation/usage/gpio/python/README.md

6.2.3 Introducere in circuitele electronice

O serie de tutoriale sunt prezentate in aceasta sectiune. Pe lângă Raspberry Pi, sunt necesare următoarele componente:

- 10 x Male-to-Female jumper wires
- 1 x Breadboard

- 3 x LED bulbs
- 6 x Resistors (between 300 and 1K Ohm). We will need 3x 1K Ohms for the distance sensor, and one 300 to 1K Ohm resistance per LED bulb
- 1 x HC-SR04 Ultrasonic Distance Sensor
- 1 x buzzer

În timp ce conectarea componentelor simple la pinii GPIO este sigură, este important să fiți atenți la modul în care conectați lucrurile. LED-urile ar trebui să aibă rezistențe pentru a limita curentul care trece prin ele. Nu utilizați componente de 5V pentru componentele de 3V3. Nu conectați motoarele direct la pinii GPIO.
Luminile LED atrag cantități diferite de energie în funcție de lumină, ceea ce poate deteriora Raspberry Pi dacă este conectat fără rezistență pentru a reduce curentul. Puteți începe cu o rezistență de rezistență mare și să o reduceți treptat până când sunteți mulțumit de intensitatea luminii LED. Majoritatea LED-urilor funcționează foarte bine cu o rezistență de 300-1K Ohm. Cu cât folosiți mai multă rezistență, cu atât LED-ul va fi mai slab. Dacă conectați un LED cu o rezistență de 300 Ohm și nu vedeți nicio lumină, atunci fie becul este prost, fie nu este conectat corect. Încercați un alt LED și verificați dacă este conectat corect, adică cablul mai lung al LED-ului este capătul pozitiv și ar trebui să fie conectat la un pin GPIO, iar cablul mai scurt este capătul negativ și ar trebui să meargă la solul circuitului.
În primul rând, dacă nu sunteți familiarizați cu o placă de calcul, ideea este să vă ajute să testați rapid un circuit, să faceți prototipuri, precum și să vă ajutați să aflați despre lucrul cu circuite fără a face nicio lipire.
Un panou tipic arată ceva de genul:

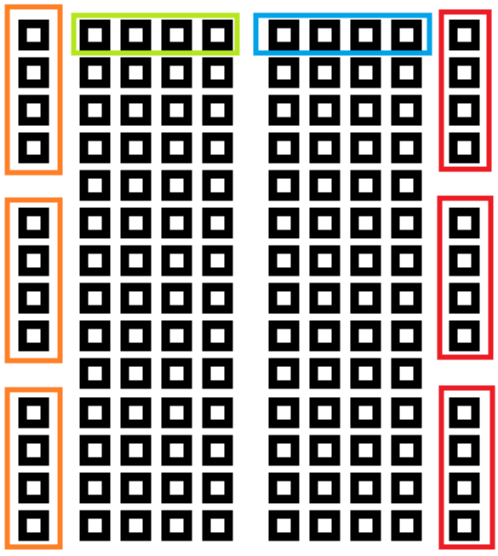


FIGURA 28 SCHEMATIC TOP VIEW OF A TYPICAL BREADBOARD

Așa cum se arată în Figura 8, rândurile de-a lungul marginilor sunt conectate vertical, iar porțiunile centrale sunt conectate orizontal. Dacă există un decalaj mai mare, ca în mijlocul

acestei plăci, nu este conectat. Dacă doriți ca aceste porțiuni să fie conectate, trebuie să le conectați cu un fir jumper.

Există panouri de diferite dimensiuni care pot fi utilizate în funcție de dimensiunea circuitului. Cele mai frecvente sunt prezentate în Figura 9 împreună cu un Raspberry Pi pentru referințe la dimensiuni.

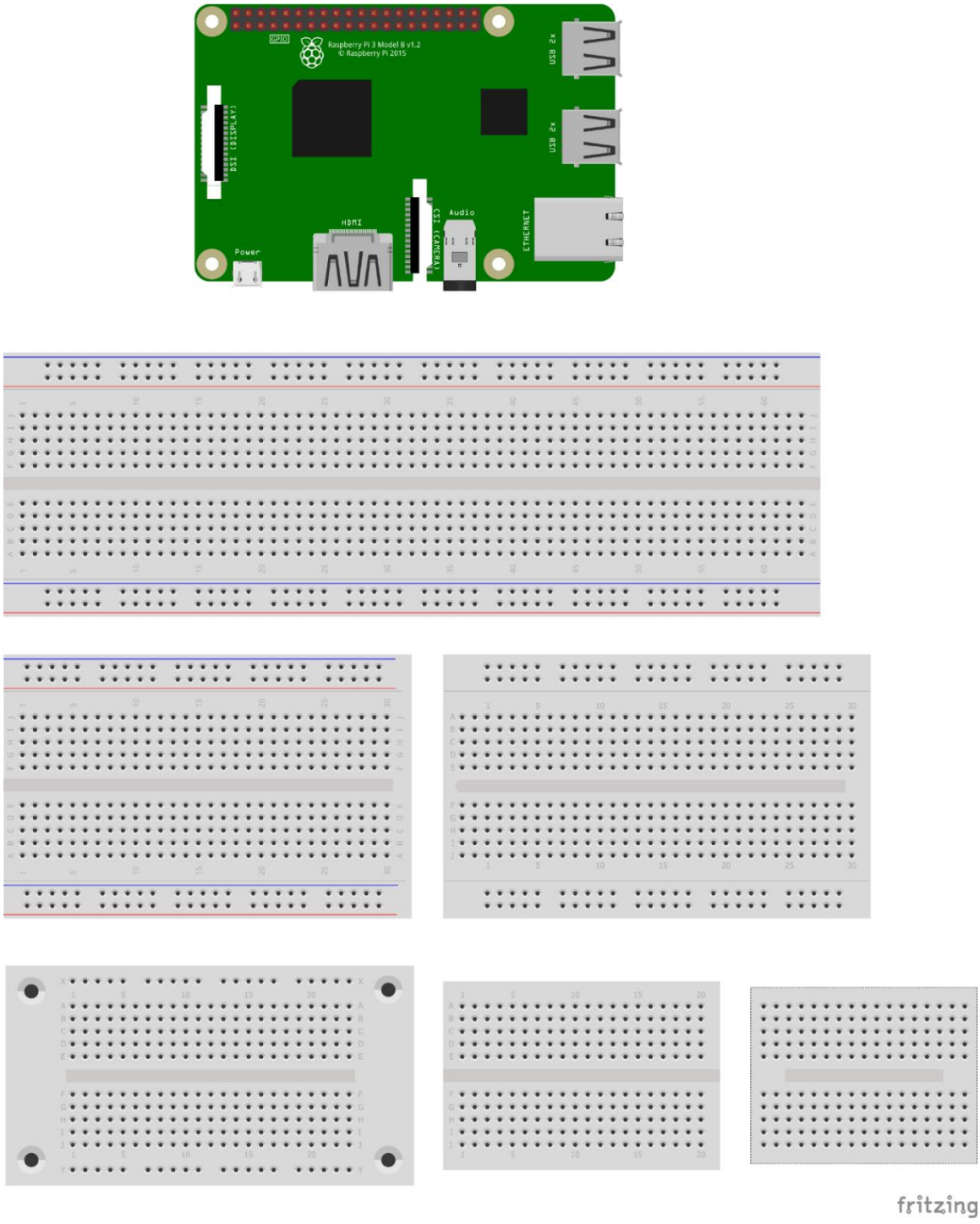


FIGURA 29 SCHEMATIC VIEW OF VARIOUS BREADBOARDS

Pe lângă componentele care au fost explicate la începutul acestei secțiuni, veți avea nevoie și de o tastatură pe care o puteți conecta la porturile USB ale Raspberry Pi.

6.2.3.1 Clipește un LED

Înainte de a continua, opriți Raspberry Pi și deconectați-l. Pentru acest circuit, veți avea nevoie de o placă de măsurare, un LED, un rezistor de 220 Ohm și două fire jumper de la femeie la mascul. Circuitul complet este în diagrama schematică, prezentată în Figura 10.

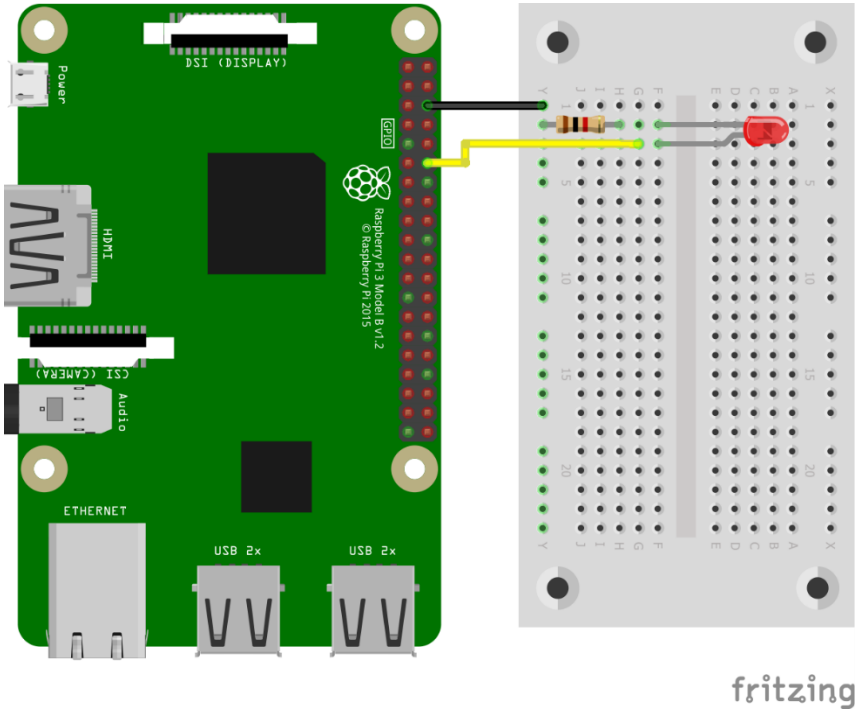


FIGURA 30 SCHEMATIC OF LED CIRCUIT

Acum puteți porni Raspberry Pi și veți începe să dezvoltați un cod simplu în Python. În majoritatea cazurilor biblioteca GPIO Python este deja instalată în sistemul de operare al Raspberry Pi. Dacă nu, atunci trebuie să-l instalați. Pentru aceasta, deschideți un terminal și tastați linia de comandă: $ sudo apt-get install python-rpi.gpio
Din meniul desktop din stânga sus selectați Programare și apoi Thonny Python (consultați capturile de ecran care urmează).

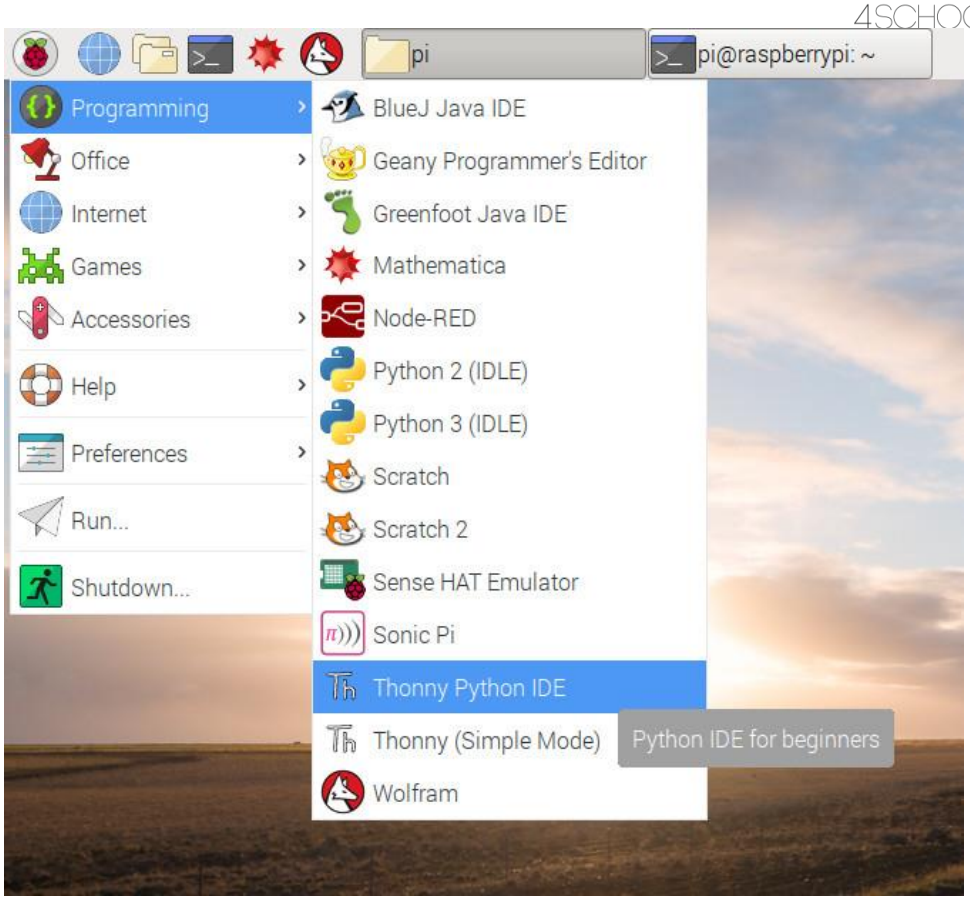


FIGURA 31 LOCATING THONNY PYTHON

Mai întâi trebuie să creați un fișier Python nou făcând clic pe Fișier File Fișier nou și salvați-l sub numele led.py.
Apoi, sunteți gata să începeți să tastați primele linii ale programului:

```
importați RPi.GPIO ca GPIO
timp de import
GPIO.setmode (GPIO.BCM)
```

Numerotarea BCM este numărul PIN pe care îl consideră cipul Broadcom. Veți utiliza numerotarea BCM, care este, de asemenea, prezentată în figurile anterioare. Este util să imprimați acest lucru și să îl aveți întotdeauna la îndemână, deoarece va trebui să verificați dacă conexiunile dvs. de circuit sunt corecte și că corespund codului pe care îl programați.
Acum, continuând scriptul de cod, ați setat:

```
GPIO.setup (18, GPIO.OUT)
```

Aici, configurați pinul pentru a fi un pin care transmite informații. De asemenea, puteți seta pinul pentru a prelua informații.

```
GPIO.output (18, GPIO.HIGH)
```

time.sleep (3)
Aici, îi spuneți pinului 18 să emită un semnal înalt, apoi să facă o pauză timp de 3 secunde.

```
GPIO.output (18, GPIO.LOW)
GPIO.cleanup ()
```

În cele din urmă, schimbați ieșirea pinului la un semnal LOW și apoi utilizați GPIO.cleanup () pentru a reseta stările pinului la stările lor implicite înainte de a termina cu programul. Acest lucru se face astfel încât următorul program care rulează intră într-o stare pe care

o așteaptă. Este destul de ușor să uitați și să lăsați diferiți pini în configurație diferit și să constatați că ulterior aceștia acționează ciudat.
Scriptul complet este după cum urmează. Diferite linii de comentarii sunt inserate pentru comoditate.The BCM numbering is the pin number that the Broadcom chip considers it to be. You will use the BCM numbering which is also shown in previous figures. It is useful to print this and to always have it somewhere handy, as you will need to check that your circuit connections are correct and that they correspond to the code you program.

```
################################################

# first import libraries and set GPIO numbering mode
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)

# setup pin of LED as output and turn it on
GPIO.setup(18, GPIO.OUT)
GPIO.output(18, GPIO.HIGH)
# sleep for 3 seconds and then turn off LED
time.sleep(3)
GPIO.output(18, GPIO.LOW)

# cleanup gpio before exiting
GPIO.cleanup()
################################################
```

Acum sunteți gata să rulați programul. Mai întâi salvați programul și apoi faceți clic pe butonul F5 pentru a executa scriptul.
Când programul rulează, LED-ul ar trebui să se aprindă timp de trei secunde și apoi ar trebui să se stingă. Puteți încerca să adăugați mai multe LED-uri în circuit și cod sau să adăugați un buzzer pentru a scoate un sunet pe care LED-urile sunt aprinse. De asemenea, puteți încerca să practicați construcții Python simple, cum ar fi bucle, pentru a face LED-urile să clipească de mai multe ori.

6.2.3.2 Activati un buzzer

Înainte de a continua cu acest tutorial, veți încerca mai întâi să utilizați un pin GPIO ca intrare. Veți avea nevoie de un buton și un circuit așa cum se arată în schema de mai jos (Figura 11). În acest caz, veți utiliza un buzzer în loc de un LED, care este conectat la pinul GPIO 18. În programul dvs. veți seta acest pin ca ieșire. De asemenea, veți conecta un buton la pinul GPIO 24 pe care îl veți configura ca intrare. Celălalt capăt al butonului ar trebui să fie conectat la pinul de tensiune de 3,3V al Raspberry Pi printr-un rezistor. Astfel, când apăsați butonul, atunci pinul GPIO 24 va fi în stare HIGH, adică în 3,3V.
Când ați terminat cu circuitul, puteți porni Raspberry Pi și puteți începe să tastați programul într-un nou fișier Python (button.py). Scriptul complet este prezentat mai jos, împreună cu mai multe linii de comentarii pentru explicații.
În programul dvs. importați mai întâi modulele Python de care aveți nevoie, apoi configurați pinii GPIO pentru buzzer și buton ca ieșire și respectiv intrare. Apoi, doar pentru a verifica, porniți soneria timp de 3 secunde. La final, începeți o buclă care rulează pentru totdeauna. În buclă verificați dacă pinul de intrare este în stare HIGH. Dacă da, atunci acest lucru înseamnă că butonul este apăsat și, prin urmare, doriți să setați pinul de ieșire

să fie HIGH pentru a porni soneria. În caz contrar, pinul de ieșire este setat la LOW, astfel încât soneria este oprită.

When you are done with your circuit, you can turn on the Raspberry Pi and start typing your program in a new Python file (`button.py`). The complete script is presented below along with several comment lines for explanations.

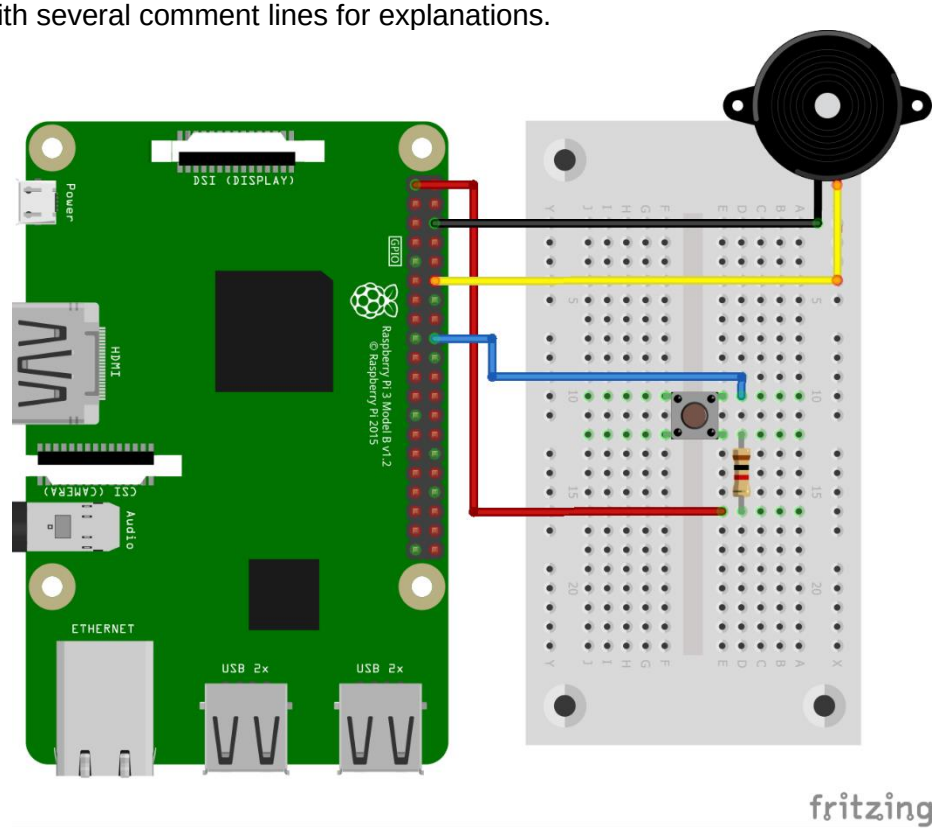


FIGURA 32 DIAGRAMA SCHEMATICA A UNUI CIRCUIT CU PUSH BUTTON ASI BUZZER CONECTAT LA GPIO PINS

```
################################################
# first import modules and GPIO numbering mode
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.cleanup()
# pin for buzzer (or led)
pin_out = 18
GPIO.setup(pin_out, GPIO.OUT)
# pin for push button
pin_button = 24
GPIO.setup(pin_button, GPIO.IN)
# turn on buzzer for 3 sec
GPIO.output(pin_out, GPIO.HIGH)
time.sleep(3)
GPIO.output(pin_out, GPIO.LOW)
# do this loop forever! Press Ctrl-C to stop it
# When button is pressed then buzzer is on
```

```
while True:
    if GPIO.input(pin_button) == GPIO.HIGH:
        print("button pushed")
        GPIO.output(pin_out, GPIO.HIGH)
    else :
        GPIO.output(pin_out, GPIO.LOW)
##############################################
```

6.2.3.3 Conectarea unui a senzor

În acest tutorial, veți utiliza un senzor, senzorul de distanță cu ultrasunete HC-SR04, împreună cu intrarea GPIO.
Senzorul de distanță HC-SR04 măsoară distanța pe baza emiterii unei explozii de sunet și a sincronizării timpului necesar pentru a primi ecoul înapoi. Folosind constanta cunoscută care este viteza sunetului, putem determina matematic distanța oricărui obiect din fața acestui senzor prin simpla măsurare a timpului care a trecut pe măsură ce undele sonore au fost emise, a lovit obiectul din fața senzorului, a revenit înapoi și a revenit la senzor.
Înainte de a continua, opriți Raspberry Pi și deconectați-l. Pentru circuitul acestui tutorial veți avea nevoie de:

- Breadboard
- 4x Male-to-Female jumper wire cables
- 1x 1K Ohm and 1x 2K Ohm resistor, or 3x 1K Ohm resistors
- 1x HC-SR04 ultrasonic distance sensor

Senzorul de distanță vine cu 4 pini: putere (VCC), declanșator (TRIG), ecou (ECHO) și masă (GND). Pinul de alimentare va fi conectat la pinul de 5V al Raspberry Pi, declanșatorul va fi atribuit unui pin GPIO ca ieșire, ecoul va fi atribuit unui pin GPIO ca intrare, iar masa va fi conectată la un pin de masă de pe Raspberry Pi.
Figura 12 prezintă configurarea completă.

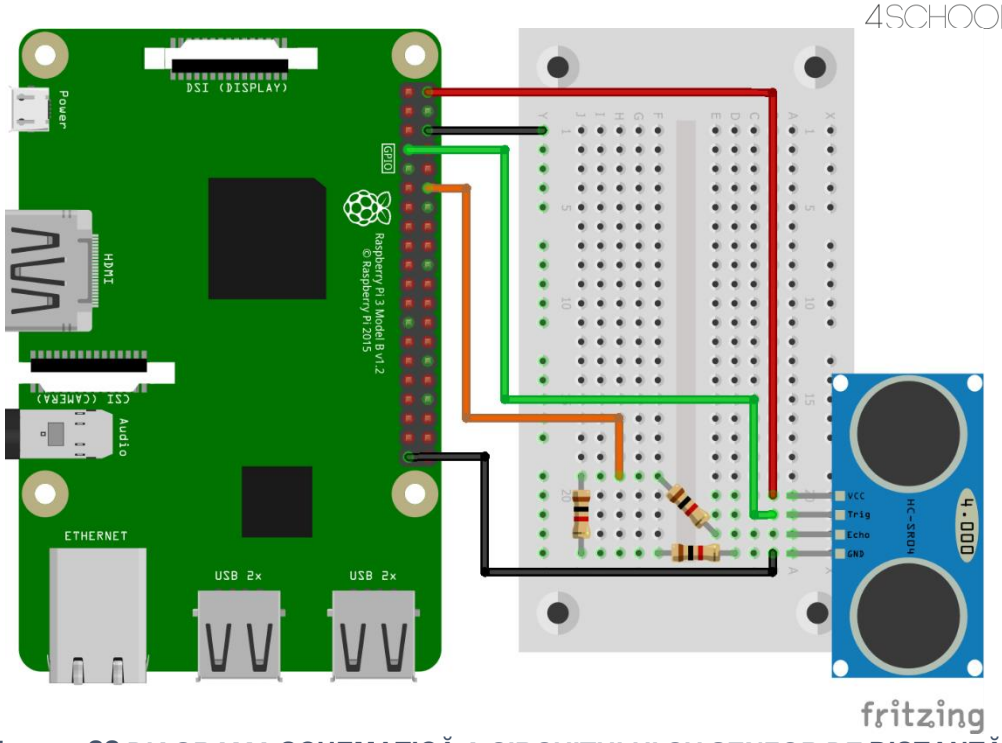


FIGURA 33 DIAGRAMA SCHEMATICĂ A CIRCUITULUI CU SENZOR DE DISTANȚĂ CONECTAT LA PINS GPIO

Acum puteți porni Raspberry Pi, puteți crea un nou fișier Python și puteți începe noul program (`sensor.py`).

La fel ca în tutorialul anterior, importați mai întâi aceleași biblioteci și modul inițial.

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
```

Apoi definiți pinii TRIG și ECHO și le configurați ca ieșire și respectiv intrare:

```
TRIG = 4
ECHO = 18
GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)
```

Aici, definiți TRIG și ECHO ca numere de pin Broadcom pe care intenționați să le utilizați pentru acea parte a senzorului. Acest lucru se face deoarece trebuie să faceți referire la ambii pini de mai multe ori.

Apoi, cu următoarele, emiteți o explozie de ultrasunete de la senzor:

```
GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)
```

Trebuie să măsurați timpul necesar pentru ca ecoul exploziei să fie detectat înapoi de către senzor:

```
while GPIO.input(ECHO) == False:
    tstart = time.time()
```

După aceea, emiteți un semnal și apoi măsurați intervalul de timp:

```
while GPIO.input(ECHO) == True:
    tend = time.time()
sig_time = tend-tstart
```

Când în sfârșit obțineți un timp de intrare, puteți scădea ora de încheiere din ora de început și puteți calcula distanța înmulțind cu viteza sunetului. Viteza sunetului în aer uscat la 20 ° C este de 343,26 m / sec.
Rețineți, de asemenea, că trebuie să vă înmulțiți cu ½, deoarece explozia de sunet parcurge de două ori distanța până la obiect sau obstacol (adică este emisă de senzor, se deplasează până când este sărit pe un obiect și se deplasează înapoi la senzor pentru a fi detectat):

```
distance = sig_time * 34326.* 0.5
print("signal(sec), distance(cm):", sig_time, distance)
GPIO.cleanup()
```

Puteți imprima distanța și apoi curățați pinii.
Ieșirea dvs. ar trebui să fie ceva de genul: semnal (sec), distanță (cm): 0,001 17,163
Rețineți că, după aproximativ 20 de grade de unghi, distanța dvs. ar putea fi foarte înclinată, deoarece săritura inițială ar putea să rateze și ar putea lua explozia mai multe sărituri de pe lucruri pentru a reveni la senzor.
Scriptul de cod complet pentru acest tutorial este după cum urmează. Diferite linii de comentarii sunt inserate pentru confort și claritate.

```
################################################
# first import libraries and set GPIO numbering mode
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)

# define TRIG and ECHO pins and set them up as output and input
TRIG = 4
ECHO = 18
GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)
# emit a burst of ultrasound
GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)

# measure time interval
while GPIO.input(ECHO) == False:
    tstart = time.time()
while GPIO.input(ECHO) == True:
    tend = time.time()
sig_time = tend-tstart

# calculate distance and print value
distance = sig_time * 34326.* 0.5
print("signal(sec), distance(cm):", sig_time, distance)

# cleanup gpio before exiting
GPIO.cleanup()
################################################
```

Puteți continua combinând circuite și coduri din acest tutorial și cel anterior pentru a adăuga unul sau mai multe LED-uri (de exemplu, vedeți figura 13 de mai jos).

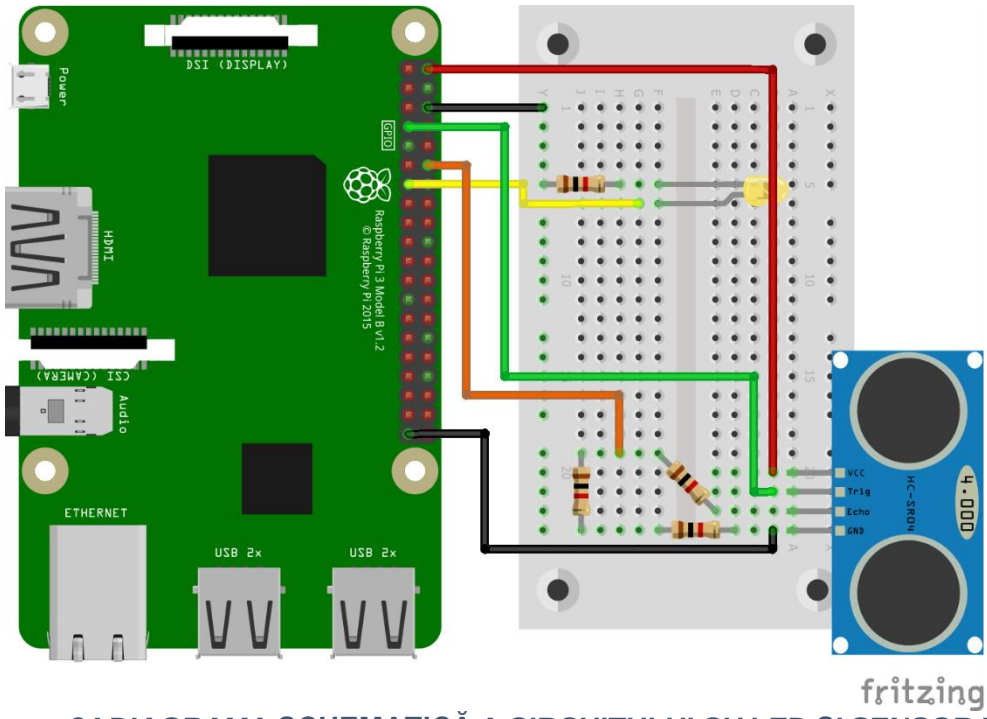


FIGURA 34 DIAGRAMA SCHEMATICĂ A CIRCUITULUI CU LED ȘI SENSOR DE DISTANȚĂ CONECTAT LA PIN-URILE GPIO

6.2.3.4 Creați o alarmă de proximitate

În acest tutorial, vom pune laolaltă ceea ce am învățat până acum pentru a crea o alarmă de proximitate sau o lumină de stop, cum ar fi cele pe care le au mașinile pentru a ajuta șoferii atunci când își parchează mașinile.
Ideea principală a unei alarme de proximitate sau a unui semnal de stop este să arate verde atunci când este suficient spațiu, să devină galben pe măsură ce distanța crește și apoi roșu când se atinge o distanță minimă, adică vehiculul ar trebui să se oprească. Vom construi acest sistem combinând circuitele și codul pe care le-am dezvoltat în tutorialele noastre anterioare.
Mai întâi, lăsăm circuitul senzorului de distanță așa cum este și adăugăm circuitele pentru cele trei lumini LED. Figura 14 prezintă configurarea completă.

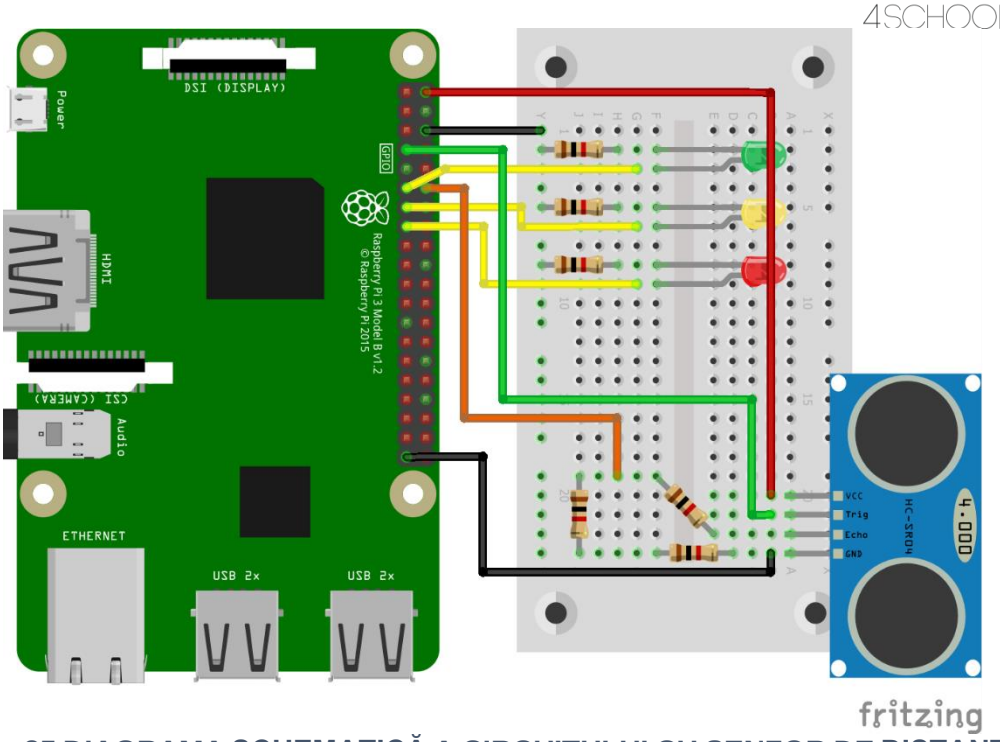


FIGURA 35 DIAGRAMA SCHEMATICĂ A CIRCUITULUI CU SENZOR DE DISTANȚĂ ȘI LED-URI MULTIPLE CONECTATE LA PIN-URILE GPIO

Odată ce totul este conectat, se recomandă să-l verificați înainte de a porni Raspberry Pi. Apoi, trebuie să modificați scriptul anterior al senzorului de distanță și să introduceți o buclă continuă și să transformați anumite părți ale codului în funcții. Puteți începe prin copierea-lipirea codului existent din tutorialul anterior într-un nou fișier Python (`prossimitate_alarm.py`). Noul cod va arăta după cum urmează:

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
# doing this first, since using a while True.
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
TRIG = 4
ECHO = 18
GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)

def get_distance():
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)
    while GPIO.input(ECHO) == False:
        tstart = time.time()
    while GPIO.input(ECHO) == True:
        tend = time.time()
    sig_time = tend-tstart
```

```
    # Distance in cm
    distance = sig_time * 34326.* 0.5
    #print("signal(sec), distance(cm):", sig_time, distance)
    return distance

while True:
    distance = get_distance()
    time.sleep(0.05)
```

Observați că ați pus cea mai mare parte a codului în funcția get_distance () și apoi îl apelați într-o buclă de timp care rulează pentru totdeauna. Rețineți, de asemenea, că trebuie să puneți ceva timp de somn între fiecare funcționare a funcției, altfel este posibil ca senzorul să nu se comporte normal. Dacă nu-l lăsați să doarmă pentru o clipă, acesta va fi doar copleșit și oprit. Acum, aveți nevoie doar de un cod simplu care să aprindă luminile LED. De exemplu, o lumină verde, presupunând că ați alimentat lumina verde la pinul BCM 17:

```
GREEN = 17
GPIO.setup(GREEN,GPIO.OUT)
def green_light():
    GPIO.output(GREEN, GPIO.HIGH)
```

Puteți copia și lipi acest lucru pentru celelalte LED-uri, dar observați că nimic nu le oprește dacă schimbăm luminile, așa că `green_light()` ai nevoie de:

```
def green_light():
    GPIO.output(GREEN, GPIO.HIGH)
    GPIO.output(YELLOW, GPIO.LOW)
    GPIO.output(RED, GPIO.LOW)
```

Acum puteți face doar copiere și lipire și faceți acest lucru pentru toate luminile:

```
GREEN = 17
YELLOW = 27
RED = 22
GPIO.setup(GREEN,GPIO.OUT)
GPIO.setup(YELLOW,GPIO.OUT)
GPIO.setup(RED,GPIO.OUT)
def green_light():
    GPIO.output(GREEN, GPIO.HIGH)
    GPIO.output(YELLOW, GPIO.LOW)
    GPIO.output(RED, GPIO.LOW)
def yellow_light():
    GPIO.output(GREEN, GPIO.LOW)
    GPIO.output(YELLOW, GPIO.HIGH)
    GPIO.output(RED, GPIO.LOW)
def red_light():
    GPIO.output(GREEN, GPIO.LOW)
    GPIO.output(YELLOW, GPIO.LOW)
    GPIO.output(RED, GPIO.HIGH)
```

Apoi, bucla dvs. while ar putea arăta ceva de genul:

```
while True:
    distance = get_distance()
    time.sleep(0.05)
    print(distance)
    if distance >= 30:
        green_light()
    elif 30 > distance > 10:
        yellow_light()
    elif distance <= 10:
        red_light()
```

Deci, dacă distanța este mai mare sau egală cu 30 cm, este afișată o lumină verde. Dacă este între 10 și 30 cm, este afișată lumina galbenă, iar apoi se afișează o lumină roșie pentru mai puțin sau egal cu 10 cm.
Scriptul de cod complet este după cum urmează. Diferite linii de comentarii sunt inserate pentru confort și claritate.

```
################################################
# first import libraries and set GPIO numbering mode
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
# doing this first, since we're using a while True.
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)

# define TRIG and ECHO pins and set them up as output and input
TRIG = 4
ECHO = 18
GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)
# function to get distance value
def get_distance():
    # emit a burst of ultrasound
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)
    # measure time interval
    while GPIO.input(ECHO) == False:
        tstart = time.time()
    while GPIO.input(ECHO) == True:
        tend = time.time()
    sig_time = tend-tstart
    # Distance in cm
    distance = sig_time * 34326.* 0.5
    #print("signal(sec), distance(cm):", sig_time, distance)


    return distance
```

```
# define pins for LEDs and setup them as outputs
GREEN = 17
YELLOW = 27
RED = 22
GPIO.setup(GREEN,GPIO.OUT)
GPIO.setup(YELLOW,GPIO.OUT)
GPIO.setup(RED,GPIO.OUT)
# function to turn on green, turn off yellow and red
def green_light():
    GPIO.output(GREEN, GPIO.HIGH)
    GPIO.output(YELLOW, GPIO.LOW)
    GPIO.output(RED, GPIO.LOW)
# function to turn on yellow, turn off green and red def
yellow_light():
    GPIO.output(GREEN, GPIO.LOW)
    GPIO.output(YELLOW, GPIO.HIGH)
    GPIO.output(RED, GPIO.LOW)
# function to turn on red, turn off yellow and green
def red_light():
    GPIO.output(GREEN, GPIO.LOW)
    GPIO.output(YELLOW, GPIO.LOW)
    GPIO.output(RED, GPIO.HIGH)

# loop that runs forever
while True:
    distance = get_distance()
    time.sleep(0.05)
    print(distance)
    # check distance and turn on light accordingly
    if distance >= 30:
        green_light()
    elif 30 > distance > 10:
        yellow_light()
    elif distance <= 10:
        red_light()
##############################################
```

Când rulați acest program, dispozitivul în funcțiune va indica mult spațiu (cu lumină verde), se apropie (lumină galbenă), se oprește (lumină roșie).

6.3 Exemple Practice

Pentru a vă testa cunoștințele și a practica mai departe, vă propunem câteva exemple practice. Acestea se recomandă a fi realizate după fiecare tutorial și includ următoarele:

- Exemplul 1: utilizați un buzzer și mai multe LED-uri cu pin Raspberry Pi GPIO
- Exemplul 2: Măsurați viteza sunetului
- Exemplul 3: Faceți o alarmă de proximitate cu lumină și sunet

6.3.1 Exemplul 1: Foloseste un Buzzer and Multiple LEDs cu Raspberry Pi GPIO Pins

În acest exercițiu, încercați să modificați circuitul și codul pe care le-ați dezvoltat în primul tutorial pentru a vă conecta și utiliza mai multe LED-uri și pentru a le programa să clipească cu un anumit tipar (de exemplu, când unul este pe celălalt este oprit sau intermitent în diferite intervale de timp). Pentru acest exercițiu, creați un circuit așa cum se arată în Figura 15. De asemenea, puteți conecta un buzzer pentru a scoate un sunet.

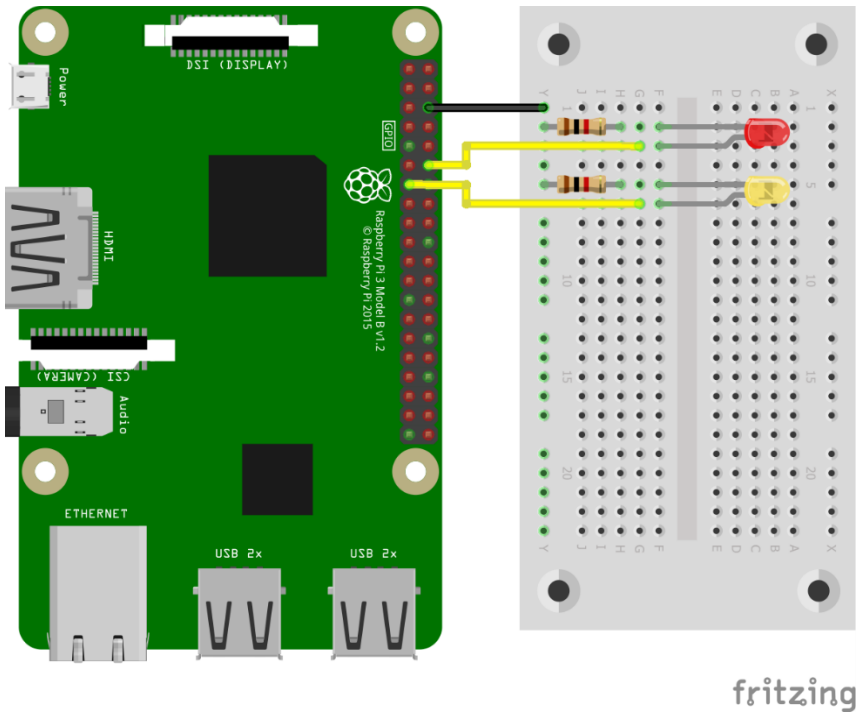


FIGURA 36 DIAGRAMA SCHEMATICĂ A CIRCUITULUI CU MULTE LED-URI CONECTATE LA PIN-URILE GPIO

6.3.2 Examplul 2: Masurarea Vitezei sunetului/ the Speed of Sound

În acest exercițiu încercați să măsurați viteza sunetului folosind senzorul și circuitul din al doilea tutorial. Pentru aceasta trebuie să modificați codul astfel încât să imprime timpul de semnal în loc de distanța calculată. Luați mai multe măsurători pentru diferite distanțe cunoscute între mână, obstacol sau obiect și senzor. Scrieți într-un tabel perechi de distanțe cunoscute (de exemplu, pe care le-ați măsurat cu o riglă) și măsurătorile de timp și apoi estimați singuri viteza sunetului.

6.3.3 Exemplul 3: Faceți o alarmă de proximitate cu lumină și sunet

În acest exercițiu, încercați să adăugați un buzzer la alarma dvs. din Tutorialul 3 pentru a face sunet în funcție de distanță. Pentru acest exercițiu puteți face un circuit așa cum se arată în Figura 16.

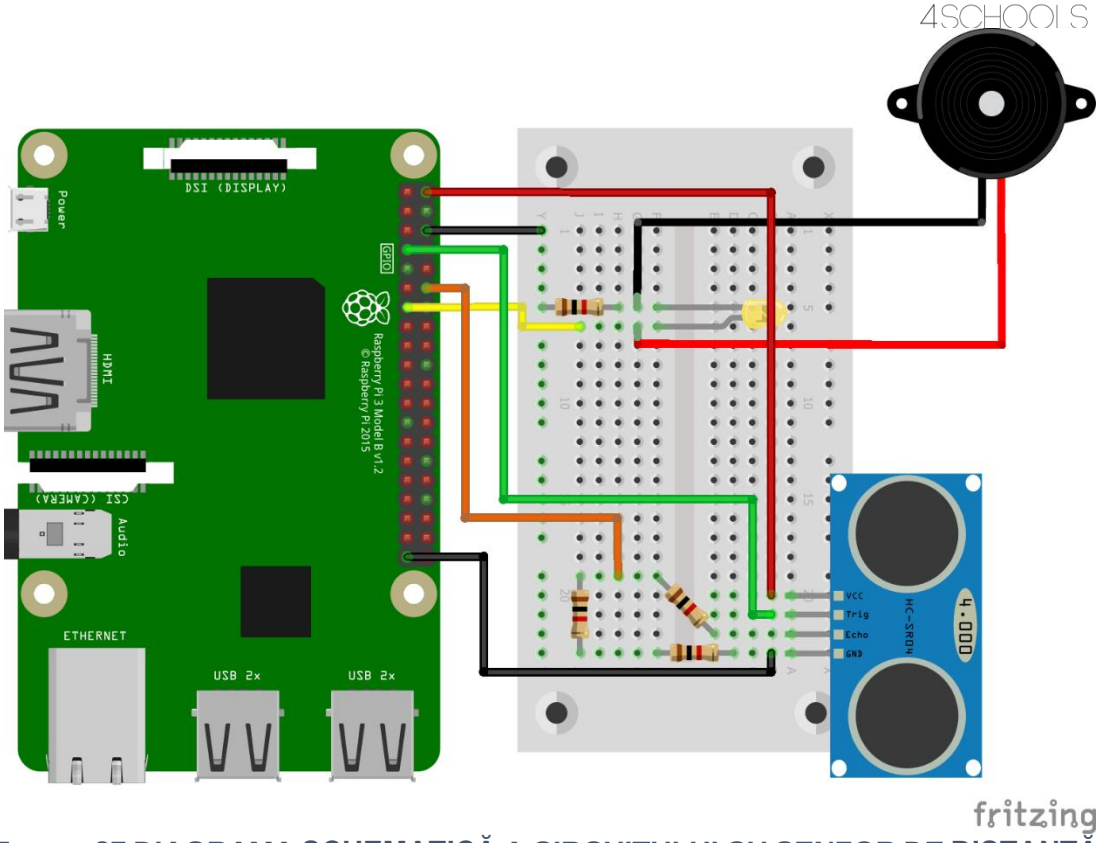


FIGURA 37 DIAGRAMA SCHEMATICĂ A CIRCUITULUI CU SENZOR DE DISTANȚĂ, LED ȘI BUZZER CONECTAT LA PINS GPIO

6.4 Test de evaluare

1. Putem configura un pin GPIO pentru a fi:
    a. Input
    b. Output
    c. Input ori Output
2. Ar trebui să opriți Raspberry Pi înainte de a utiliza pinii GPIO.
    a. Adevarat
    b. Fals
3. În ce stare poate fi un pin GPIO?
    a. Inalt
    b. Inalt sau jos
    c. Jos
4. GPIO-ul Raspberry Pi are pini de tensiune de 3,3V și 10V.
    a. Adevarat
    b. Fals
5. Unul dintre cei mai importanți pini ai antetului cu 40 de pini sunt::
    a. Știfturile de la sol
    b. Pinii de tensiune
    c. Pinii GPIO
6. Folosind programarea Python și pinii GPIO puteți interacționa cu lumea fizică.
    a. Adevarat
    b. Fals
7. Folosind Python în Raspberry Pi putem programa:
    a. pini de masă
    b. pinii de tensiune
    c. Pinii GPIO
8. Conectarea GPIO cu o placă de măsurare avem nevoie de fire jumper de la bărbat la bărbat.
    a. Adevarat
    b. Fals
9. Python este:
    a. Un limbaj de programare de uz general
    b. Un limbaj de programare numai pentru Raspberry Pi
    c. Un șarpe cu picioare
10. Raspberry Pi este un computer complet funcțional, de mărime credit.
    a. Adevarat
    b. Fals

6.5 Referinte

- https://www.raspberrypi.org/documentation/usage/gpio/
- https://github.com/splitbrain/rpibplusleaf
- https://www.raspberrypi.org/documentation/usage/gpio/python/README.md
- https://www.raspberrypi.org/documentation/usage/python/README.md
- https://pythonprogramming.net/gpio-raspberry-pi-tutorials/

6.6 Resurse aditionale

- Raspberry Pi GPIO: https://www.raspberrypi.org/documentation/usage/gpio/
- Physical Computing with Python: https://projects.raspberrypi.org/en/projects/physical-computing
- Getting started with Python programming and the Raspberry Pi: https://raspberrypihq.com/getting-started-with-python-programming-and-the-raspberry-pi/
- Python Tutorial: https://docs.python.org/3/tutorial/
- MagPi magazine: https://www.raspberrypi.org/magpi/

6.7 Concluzii

Puncte cheie de concluzie pentru **Modulul 4 - Raspberry Pi programare folosind Python**:
După ce ați citit acest capitol, vă așteptați să fi învățat, înțeles și practicat următoarele elemente de învățare:

- Ce este un pin Raspberry Pi GPIO și funcționalitatea sa de bază.
- Cum să realizați circuite de bază cu LED-uri, buzzere, senzori și cum să le conectați la pinii GPIO Raspberry Pi.
- Cum se configurează, se controlează și se programează pinii GPIO utilizând limbajul Python.
- Familiarizarea cu și utilizarea principalelor construcții și sintaxei limbajului Python, inclusiv funcții, variabile, pentru bucle.

7 Calcul Fizic [P1-ECAM]

7.1 Glosar de termeni

Term / Concept	Definitie / Explicatii
Calcul fizic	Calculul fizic înseamnă interacțiunea cu obiecte din lumea reală prin programarea lor de pe un computer. Exemplele includ programarea unui LED pentru a clipi, citirea datelor de mediu de la un senzor sau chiar controlarea obiectelor robotizate. Aplicații ca acestea sunt în jurul nostru în viața de zi cu zi, de la semnale de trafic și bariere de bilete la mașini fără șofer și linii de asamblare. În spatele fiecăreia dintre aceste aplicații se află algoritmi și programe care le guvernează comportamentul. Calculul fizic combină hardware și software pentru a crea ceva util sau productiv sau pur și simplu doar pentru distracție.
Aplicatie	O aplicație este un set de coduri concepute pentru a permite îndeplinirea unor sarcini specifice. Microsoft Windows și Internet Explorer sunt exemple obișnuite.
API	API se referă la interfața de programare a aplicațiilor. Este platforma utilizată de un program pentru a accesa diferite servicii de pe sistemul informatic.
Periferic	Orice dispozitiv atașat la un computer, dar care nu face parte din acesta.
Programator	Orice echipament electronic care aranjează software scris pentru a configura circuite integrate nevolatile programabile (numite dispozitive programabile) precum EPROM-uri, EEPROM-uri, Flash-uri, eMMC, MRAM, FRAM, NV RAM, PAL-uri, FPGA-uri sau circuite logice programabile.

7.2 Continut

În acest capitol sunt descrise caracteristicile definitorii ale calculelor fizice. Vor fi identificate competențele cheie care trebuie dobândite cu calculul fizic.

7.2.1 Introducere in Calcul fizic

„Calcul fizic: detectarea și controlul lumii fizice cu ajutorul computerelor” (T.Igoe, Tisch NYU).

Aceasta este o mișcare recentă care necesită abilități în electronică, calculatoare, prelucrarea datelor, fizică și chiar conceptul de proiectare. Scopul calculului fizic este de a crea noi dispozitive în interacțiune cu lumea reală.
În acest capitol, va fi introdus acest concept de calcul fizic, iar participanților li se vor învăța aspectele teoretice și practice ale acestuia.
Acest capitol este construit în jurul unor exemple ca dezvoltarea unui robot mobil autonom, capabil să se deplaseze datorită diferiților senzori instalați pe acesta și cu privire la mai multe reguli.

Definitia din Wikipedia - free encyclopaedia:
„Calculul fizic implică sisteme interactive care pot simți și răspunde la lumea din jur. Deși această definiție este suficient de largă pentru a cuprinde sisteme precum sistemele inteligente de control al traficului auto sau procesele de automatizare din fabrică, nu este utilizată în mod obișnuit pentru a le descrie. Într-un sens mai larg, calculul fizic este un cadru creativ pentru înțelegerea relației ființelor umane cu lumea digitală. În utilizare practică, termenul descrie cel mai adesea proiecte de artă manuală, design sau DIY hobby care folosesc senzori și microcontrolere pentru a traduce intrarea analogică într-un sistem software și / sau pentru a controla dispozitivele electromecanice, cum ar fi motoarele, servomotoarele, iluminatul sau alt hardware."

Ce este calculul fizic?
Calculul fizic este o abordare a învățării modului în care oamenii comunică prin intermediul computerelor, care începe prin a lua în considerare modul în care oamenii se exprimă fizic. O mulțime de instrucțiuni inițiale de proiectare a interfeței computerului iau în calcul hardware-ul computerului - și anume, că există o tastatură, un ecran, poate difuzoare și un mouse - și se concentrează pe predarea software-ului necesar pentru a proiecta în aceste limite. În calculul fizic, luăm corpul uman ca pe un dat și încercăm să proiectăm în limitele expresiei sale.
Aceasta înseamnă că trebuie să învățăm cum un computer transformă schimbările de energie emise de corpurile noastre, sub formă de căldură, lumină, sunet și așa mai departe, în schimbarea semnalelor electronice pe care le poate citi interpretând. Aflăm despre senzorii care fac acest lucru și despre computerele foarte simple, numite microcontrolere, care citesc senzorii și le convertesc ieșirea în date. În cele din urmă, aflăm cum comunică microcontrolerele cu alte computere.

Calculul fizic înseamnă crearea sau utilizarea dispozitivelor care interacționează cu lumea din jur. Un computer fizic își simte mediul, procesează acele informații și apoi efectuează o acțiune. Acest ciclu „simț - gândește - acționează" poate fi folosit și pentru a defini un robot. La BirdBrain Technologies folosim în mod interschimbabil termenii „calcul fizic" și „robotică".
Crearea unui robot implică inginerie. Apoi, trebuie să scrieți și un program de computer pentru a face robotul să facă ceva interesant! Roboții nu arată întotdeauna ca cei arătați în filme. Unele sunt umanoizi sau mașini robotizate, dar puteți face și animale de companie robotizate sau grădini. Cerul este limita! Roboții oferă elevilor o modalitate de a-și vedea codul funcționând în lume.

Calculul fizic acoperă proiectarea și realizarea de obiecte și instalații interactive și le permite elevilor să dezvolte produse concrete, tangibile ale lumii reale, care decurg din imaginația cursanților. În acest fel, învățarea constructionistă este ridicată la un nivel care permite elevilor să câștige experiență faptică și, prin urmare, să concretizeze virtualul.

Calculul fizic are o abordare practică, ceea ce înseamnă că petreceți mult timp construind circuite, lipind, scriind programe, construind structuri pentru a ține senzori și comenzi și să aflați cum să faceți ca toate aceste lucruri să se raporteze la expresia unei persoane.

Ce trebuie să știți înainte de a lua o clasă de calcul fizic?

În primul rând, cea mai mare parte a muncii reale se întâmplă în afara orelor de curs, atât în clădirea magazinului, cât și în programare, precum și în lumea de zi cu zi, urmărind oamenii și aflând ce fac, care vă interesează să simțiți și să interpretați.
În al doilea rând, dacă nu ați programat niciodată un computer până acum, va trebui să știți puțin despre asta. În special, vă ajută dacă ați făcut o anumită programare a interfeței grafice, deoarece mai târziu în curs, învățăm cum să controlați grafica și sunetul de pe ecran cu ieșirea de la senzori. Cel puțin, ar trebui să știți și să vă simțiți confortabil cu următoarele:

- Ce este o declarație if și cum să o folosiți.
- Ce variabile sunt într-un program de computer și cum sunt utilizate
- Ce este o buclă de repetare și cum să le utilizați. Dacă vă simțiți confortabil cu afirmații cum ar fi pentru-următor, în timp ce-wend, sau repetați în timp ce ... -fin repet, atunci sunteți într-o formă bună

Cei Trei piloni de calcul fizic
Calculul fizic este un concept destul de nou și necunoscut în rândul profesorilor de informatică și al cercetătorilor în domeniul informaticii. Revizuirea literaturii a arătat că diferiți autori pun accent diferite pe următorii trei piloni ai calculului fizic: produsele, instrumentele și procesele tipice de calcul fizic trebuie să fie investigate pentru a clarifica semnificația. Nu va fi posibil să trasați clar linii între cele trei câmpuri. Procesele implică instrumente și vizează anumite produse, astfel încât perspective diferite vor pune, așadar, un accent mai puternic pe oricare dintre cele trei variabile, dar nu le vor neglija pe celelalte două. Ca punct de plecare, se presupune că calculul fizic înseamnă proiectarea creativă de obiecte sau sisteme interactive tangibile folosind hardware programabil.

7.2.2 Competente cheie in Calculul fizic

Înțelegerea sistemelor de calcul
Obiectele interactive sau instalațiile realizate cu calcul fizic sunt sisteme de calcul întregi care conțin componente hard și software pe care elevii le pot asambla și investiga în continuare. În funcție de nivelul de complexitate pe care îl suferă în cadrul specific, pot veni de la o înțelegere intuitivă (de exemplu, atunci când controlează o jucărie programabilă) la o înțelegere profundă a sistemelor interactive de calcul (de exemplu, atunci când construiește o cutie inteligentă). În special aspectele legate de proiectarea hardware îi ajută să dezvolte abilitățile de a identifica și înțelege sistemele interactive în mediile lor de zi cu zi.

Probleme de formulare
Cu calculul fizic, capacitatea de bază de a formula cu precizie probleme este formată și practică ca un prim pas în procesul de proiectare și creare a obiectelor interactive. Elevilor li se cere să descrie fără ambiguitate ceea ce se presupune că se va întâmpla dintr-o perspectivă exterioară, astfel se concentrează pe formularea problemei separat de gândirea la posibile modalități de rezolvare a problemelor.

Organizarea și analizarea datelor
În proiectele de calcul fizic, datele pot fi colectate automat cu stații meteo auto-realizate, un sistem de vot pentru a alege reprezentantul clasei următoare sau un înregistrator automat de trafic pentru a număra numărul de mașini care trec în afara școlii. În acest fel,

elevii învață cu date din lumea reală colectate în propriul mediu prin obiecte de măsurare pe care le-au proiectat și construit singuri. Aceștia vor afla despre codificarea și decodificarea datelor și informațiilor în timp ce lucrează cu senzori care furnizează date, care trebuie interpretate și actuatori care primesc date, care trebuie generate din informații.

Gândirea algoritmică

Gândirea algoritmică este, de asemenea, un element crucial al procesului de calcul fizic. Elevilor de la orice nivel li se cere să descrie cu precizie serii de evenimente, atât seriale, cât și paralele. Calculul fizic cere în special elevilor să dezvolte algoritmi care să le permită obiectelor să ruleze continuu și să interacționeze constant cu mediul.

Eficacitate și eficiență

Aspectele cheie ale gândirii computaționale includ identificarea, analiza și implementarea soluțiilor posibile cu scopul de a realiza cea mai eficientă și eficientă combinație de pași și resurse. În proiectele de calcul fizic, soluțiile ineficiente sau ineficiente sunt deosebit de evidente. Obiectele sau instalațiile interactive ar trebui să ofere feedback imediat, pot include procese simultane și ar trebui să includă interfețe intuitive. Dacă nu reușesc să îndeplinească așteptările, de ex. datorită alegerii senzorilor necorespunzători, a pauzelor excesive sau a răspunsului întârziat, acest lucru se observă imediat.

7.3 Example Practice

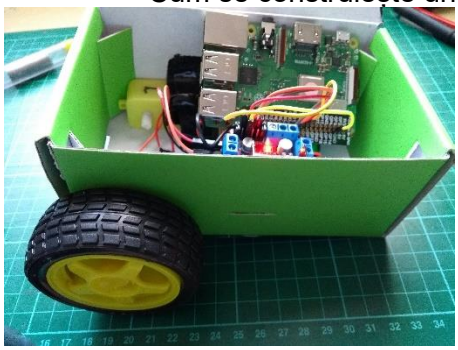
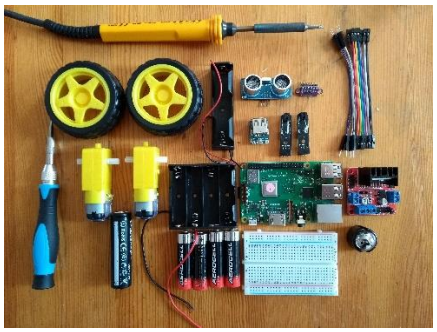
Pentru a vă testa cunoștințele și a practica mai departe, vă propunem câteva exemple practice. Acestea se recomandă a fi realizate după fiecare tutorial și includ următoarele:

- Exemplul 1: **Robot Buggy (A Raspberry Pi on wheels?)**
- Exemplul 2: **YouTube Boombox (Raspberry Pi-powered lo-fi video player for YouTube)**

7.3.1 Exemplul 1

Acest proiect pentru începători vă învață cum să configurați o placă de control a motorului, să construiți un șasiu și să utilizați Python pentru a controla motoarele. Nu vă fie teamă, este foarte ușor de configurat. În acest proiect veți construi un buggy robot pe care îl puteți programa pentru a vă deplasa folosind comenzi simple Python.

- Cum se configurează o placă de control a motorului cu două motoare
- Cum se controlează motoarele folosind Python
- Cum se construiește un șasiu robot

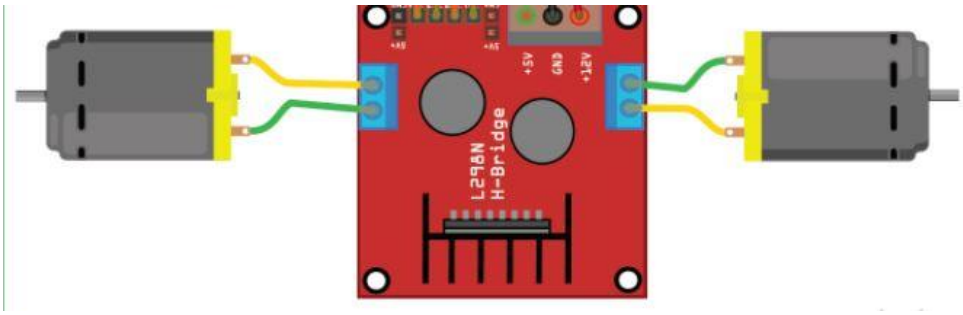
De ce ai nevoie:

- Raspberry Pi 3
- Placa controlerului motorului
- Motoare 2 × 3V - 6V DC
- 2 × roți
- 1 × suport baterie AA (pentru 4 baterii AA)
- 4 × baterii AA
- Rola cu bile
- Sârmă sau cabluri jumper
- Un pachet de baterii USB
- Șurubelniță
- Fier de lipit
- Dispozitive de decojire a firelor
- Cutie mică de carton sau plastic și lipici / bandă

Asamblarea motoarelor și a plăcii
Primul lucru pe care veți dori să îl faceți este să vă conectați placa controlerului motorului la Raspberry Pi, acumulatorul și cele două motoare, pentru a testa că toate funcționează.

Instrucțiunile de aici sunt pentru placa controlerului driverului motorului pas cu pas DC L298N Dual H Bridge și vor fi destul de similare pentru majoritatea plăcilor controlerului motorului.

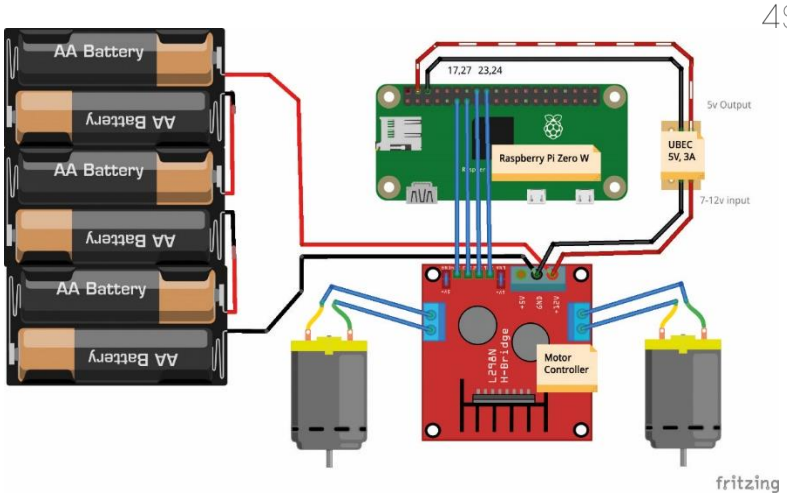
Conectați motoarele la placa
Va trebui să conectați motoarele la placă. Pentru aceasta veți avea nevoie de o șurubelniță mică.



Alimentarea motoarelor
Motoarele necesită mai multă putere decât poate oferi Raspberry Pi. Prin urmare, veți folosi patru baterii AA pentru a le alimenta.

Conectarea plăcii la Raspberry Pi
Placa utilizată în acest proiect trebuie să fie conectată la Raspberry Pi. Alte plăci se pot conecta diferit, iar unele plăci pot fi pur și simplu plasate pe pinii Raspberry Pi GPIO sub formă de HAT.

Pe placa utilizată aici există pini etichetați In1, In2, In3 și In4, precum și doi pini GND. Ce pini GPIO pe Pi pe care îl folosiți depind de dvs.; în acest proiect, GPIO 17, 27, 23 și 24 au fost utilizate.



Stânga, dreapta, înainte, înapoi
Trebuie să-ți dai seama care este motorul tău stâng și care este motorul tău drept. De asemenea, trebuie să știți ce direcție conduc pentru a merge înainte și ce direcție conduc pentru a merge înapoi.

Asamblează-ți robotul
Nu există o modalitate „corectă” de a vă construi prototipul șasiului robot, dar trebuie să țineți cont de câteva lucruri:

Șasiul trebuie să găzduiască Raspberry Pi, controlerul motorului și bateriile.
Șasiul trebuie să permită montarea unei perechi de roți.
Poate doriți să adăugați mai târziu la șasiu câteva senzori de linie și un senzor de distanță cu ultrasunete sau un senzor lidar.
Este întotdeauna o idee bună să construiești mai întâi un șasiu prototip. În cele din urmă, puteți afla cum să tăiați cu laser sau să imprimați 3D un șasiu, dar în acest proiect, o cutie de carton este utilizată ca soluție temporară.

Pentru a utiliza Raspberry Pi fără a conecta un mouse, monitor sau tastatură, îl puteți accesa de la distanță prin SSH sau VNC.
Acum puteți scrie un program pentru a vă controla robotul și a-l face să facă orice număr de lucruri.
Iată un script simplu pentru ao face să pătrundă într-o formă pătrată (poate fi necesar să schimbați ușor funcțiile de pauza):

```
from gpiozero import Robot
from time import sleep
robot = Robot(left = (17, 27), right = (23, 24))
while True:
    robot.forward()
    sleep(3)
    robot.stop()
    robot.right()
    sleep(1)
    robot.stop()
```

Acum este ocazia ta de a-ți programa robotul!

Încercați și finalizați una dintre următoarele provocări:

1. Faceți robotul să conducă într-un cerc perfect
2. Faceți robotul să conducă în zigzag
3. Faceți un mic labirint din obiecte de uz casnic și programați robotul pentru a-l naviga

Nu uitați, există doar cinci comenzi de bază pentru a vă deplasa robotul!

```
robot.forward()
robot.backward()
robot.right()
robot.left()
robot.stop()
```

7.3.2 Exemplul 2

Acest mic dispozitiv poate arăta ca partea din spate a unei camere compacte, dar este de fapt un player video lo-fi alimentat de Raspberry Pi pentru YouTube. Carcasa imprimată 3D are, de asemenea, un difuzor încorporat și un afișaj mic. Pi este setat în modul chioșc și redă automat orice flux de muzică YouTube. Un joystick încorporat poate fi folosit pentru a schimba stațiile și a regla volumul.

- BrainCraft HAT are tot ce aveți nevoie pentru a crea un player YouTube all-in-one.
- Joystick-ul încorporat poate fi folosit pentru a schimba posturile și a regla volumul.
- Acest lucru vă permite să comutați rapid între diferite canale YouTube fără a fi nevoie să utilizați tastatura sau mouse-ul.
- De asemenea, puteți întrerupe și reda videoclipul apăsând butonul de lângă joystick.

De ce avem nevoie:

Partea Electronica:

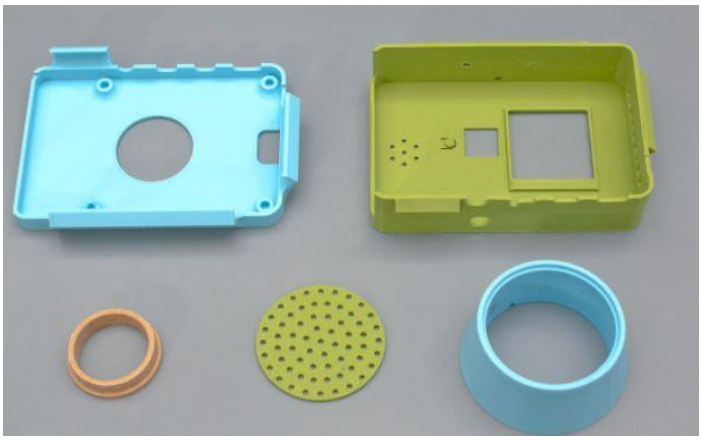
- Raspberry Pi 4 Model B - 4 GB RAM
- 16GB Card with NOOBS 3.1 for Raspberry Pi Computers including 4
- Speaker - 40mm Diameter - 4 Ohm 3 Watt
- JST PH 2-Pin Cable - Female Connector 100mm
- Full Size Wireless Keyboard with Trackpad

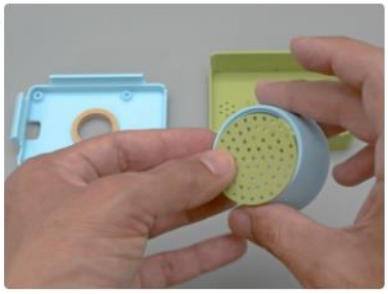
3D Printing:

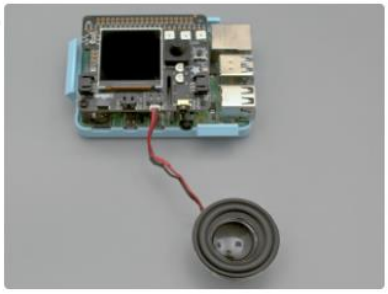
Fișierele STL pentru imprimarea 3D sunt orientate pentru a imprima „așa cum este” pe mașinile în stil FDM. Părțile sunt concepute pentru imprimarea 3D fără niciun material de sprijin. Sursa de proiectare originală poate fi descărcată folosind linkurile de mai jos.

- BrainTube-screen.stl
- BrainTube-case.stl
- BrainTube-tripod.stl
- BrainTube-speaker-grill.stl
- BrainTube-speaker-cone.stl
- BrainTube-speaker-ring.stl

https://www.thingiverse.com/thing:4739213



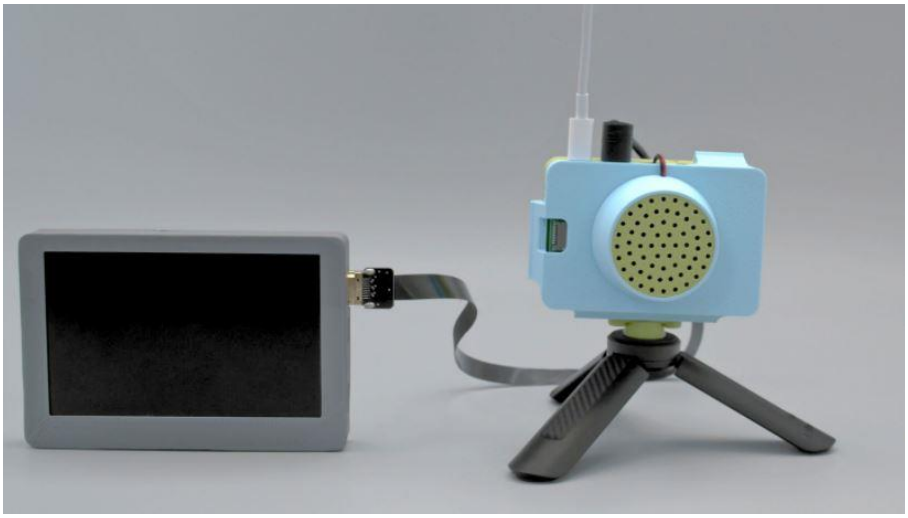




Asamblul

- Apasa = Press Fit Speaker Holder
- Piesa inelului difuzor se încadrează în decupajul piesei BrainTube-Case.
- Speaker Grill
- Introduceți partea Speaker-Grill în partea Speaker-Cone.

- • Adăugați Raspberry Pi în carcasă
- Așezați Pi deasupra standurilor și aliniați placa cu deschiderile portului de pe carcasă.
  - • Difuzor
- Lipiți firele difuzorului la conectorul feminin JST. Folosiți termocontractiv pentru a izola conexiunile
  - • Carcasa ecranului
- Capacul frontal este montat peste HAT cu cablul difuzorului montat prin fanta de pe deschiderea portului HDMI.
  - ◦ Adăugați bandă gaffers sau bandă kapton pentru a izola magnetul difuzorului. Presa Speaker se potrivește în ring.
- Aliniați conul difuzorului peste firele difuzoarelor și apoi apăsați fit pe difuzor.



Configurati Raspberry Pi

- Configurați cardul SD
  - ◦ Atașați un monitor HDMI, pentru a configura sistemul de operare Pi. Utilizați Raspberry Pi Imager oficial pentru a arde cel mai recent sistem de operare pe un card SD.
- Activați SSH
  - ◦ După finalizarea tuturor actualizărilor, activați SSH
- Instalați Blinka Libraries
  - ◦ Permite multor biblioteci care au fost scrise pentru CircuitPython să ruleze pe CPython pentru Linux. Urmați pasul de mai jos pentru a configura bibliotecile Blinka: Configurare Blinka
- Configurați sunetul, ventilatorul și afișajulConfigure Audio, Fan and Display

- Instalați scriptul în modul Kiosk
  - Utilizați scriptul în modul chioșc pentru a activa redarea video pe ecran complet
- Editați lista de redare
  - Adăugați propriile fluxuri:Install Kiosk-mode Script

7.4 Test de evaluare

1. Ce este un dispozitiv de intrare sau ieșire?
   - un computer precum iPad, smartphone, laptop sau PC
   - **o mașină mică atașată la un computer care face o treabă**
   - un tip de vehicul, cum ar fi o mașină sau un camion
   - ceva pe care l-ați descărca și instala pe iPhone

2. Ce sunt datele?
   - locul în care informațiile sunt salvate într-un computer
   - o mică parte a unui computer
   - un tip de robot
   - **informațiile utilizate de computere**

3. Tastatura este unul dintre cele mai proeminente _____ dispozitive ale unui computer.
   - **Input/ intrari**
   - Output/ iesiri
   - input si output/ intrări și ieșiri

4. Când o fotocopiator imprimă ceva, acesta _____.
   - intrări
   - **ieșiri**
   - intrări și ieșiri

5. O cască VR este un exemplu de _____ pentru un computer.
   - intrări
   - ieșiri
   - **intrări și ieșiri**

6. In Python, ce NU poți face în Shell?
- **salvați codul și rulați-l din nou**
- rulați codul
- scoateți răspunsurile la probleme de matematică

7. Care dintre acestea este un avantaj al utilizării variabilelor? (Alegeți toate răspunsurile corecte).
- **poate fi folosit pentru a stoca intrări**
- **nu trebuie să introduceți informații din nou și din nou**
- **face codul nostru mai scurt și mai simplu de citit**
- **informații mai ușor de utilizat în cod**

8. Expedierea textului. Care este corect?
- print("Happy New Year!)
- output("Happy New Year!")
- Print("Happy New Year!")
- **print("Happy New Year!")**

9. Care este modul corect de a scrie „împarte a la 2 și înmulțește cu b” în Python?
• a / 2 x b
• a \ 2 x b
• **a / 2 * b**
• a \ 2 * b

10. Cum facem o variabilă numită d care conține textul *Bună dimineața* în Python?
- d (Bună dimineața)
- d = (Bună dimineața)
- d = Bună dimineața
- **d = "Bună dimineața"**

7.5 Referinte

- https://en.wikipedia.org/wiki/Physical_computing
- https://www.tigoe.com/blog/what-is-physical-computing/
- https://www.qaeducation.co.uk/news/physical-computing#:~:text=Physical%20computing%20means%20interacting%20with,or%20even%20controlling%20robotic%20objects.
- https://guides.temple.edu/c.php?g=419841&p=2863656
- Mareen Przybylla, Ralf Romeike, Key Competences with Physical Computing https://publishup.uni-potsdam.de/opus4-ubp/frontdoor/deliver/index/docId/8290/file/cid07_S351-361.pdf
- https://sunnie-sva-physicalcomputing.tumblr.com/post/66904922347/physical-computing-mid-term-project-music
- https://quizizz.com/admin/quiz/5e153511ae952c001b88401a/physical-computing-review-quiz-1-d

7.6 Resurse aditionale

- Filiz Kalelioglu, Sue Sentence, (2020) Teaching with physical computing in school: the case of the micro:bit, *Education and Information Technologies* 25, 2577–2603.
- Tom Igoe, Making Things Talk: Practical Methods for Connecting Physical Objects, Make; 1 edition (September 28, 2007), ISBN-10: 0596510519 (source of examples). Second Edition, Released: August 2011 (est.) ISBN-10: 1449392431, ISBN-13: 978-1449392437, http://oreilly.com/catalog/0636920010920
- Joshua Noble, Programming Interactivity: A Designer's Guide to Processing, Arduino, and OpenFrameworks, O'Reilly Media, July 2009
- Alvaro Cassinelli, Daniel Saakes, Data Flow, Spatial Physical Computing, TEI 2017, March 20–23, 2017, Yokohama, Japan

7.7 Concluzii

Calculul fizic este o activitate complexă care solicită cursanților să fie conștienți de problemele software și hard în același timp. La nivel introductiv (școala primară), elevii pot lucra cu jucării programabile sau cărămizi programabile și medii de programare drag & drop pentru a învăța fundamentele gândirii algoritmice. Kituri de construcție care au senzori și actuatori preasamblați pentru a fi conectați la un microcontroler fie direct, fie cu un ecran permite copiilor mai mari să ajungă foarte repede la realizări vizibile și tangibile. Odată cu progresul în calcul fizic, atât pe partea hard, cât și pe cea software, elevii sunt supuși unor procese de învățare care întăresc gândirea computațională și competențele cheie care sunt necesare pentru toate aspectele vieții. Calculul fizic poate îmbogăți viitoarele săli de informatică cu competențe valoroase care se concentrează în educația informatică mai bine decât în multe alte discipline, deoarece sunt încluse în materie și nu trebuie impuse artificial. Cercetările viitoare pe această temă vor investiga, prin urmare, modul în care elevii din diferite grupe de vârstă experimentează activități fizice de calcul și procese de învățare specifice.

8 Concluzii Generale

Chiar dacă copiii se nasc în tehnologie, par să aibă nevoie de abilități tehnologice, cum ar fi programarea. Sunt necesare noi modalități de implicare a copiilor în programare și STEM. Alegem jocul hands-on prin crearea de jocuri care pot fi redate pe un computer DIY din lemn cu design retro, în combinație cu gadget-uri electronice legate de subiecte STEM. Această abordare este distractivă și mai educativă în loc de mai mult timp pe ecran. Conectarea lumilor online și offline poate oferi copiilor un mediu mai captivant și mai sănătos pentru a învăța cum să programeze și să dezvolte abilități STEM.

Obiectivul principal al proiectului Erasmus+ STEMKIT4Schools este de a produce abordări și instrumente care să îi ajute pe cei care lucrează cu copii prin dezvoltarea abilităților legate de STEM. Acesta își propune să realizeze acest lucru nu prin creșterea timpului pe ecran, ci prin încurajarea jocului practic.

Un curriculum complet a fost conceput de către partenerii proiectului pentru a răspunde obiectivelor STEMKIT4Schools, inclusiv planuri de lecții pentru utilizarea Minecraft Pi / Scratch / Python / Kits cu computerul STEMKIT în clasă. Planurile de lecție fac parte din ghidul educatorului pentru profesori. Kituri electronice sunt proiectate și construite pentru a completa predarea de programare, calcul fizic și subiecte STEM. Cursul final STEMKIT predă elemente de bază ale jocurilor cu Minecraft Pi, Python, Scratch, precum și calcul fizic (folosind componente digitale, analogice și electromecanice de bază) și colaborare (interacțiune și partajare cu ceilalți). Computerul DIY STEMKIT se bazează pe Raspberry Pi, ceea ce înseamnă că poate valorifica puterea și resursele practic nelimitate pentru Raspberry, în timp ce caracteristicile sistemului de operare Raspbian oferă capacități suplimentare. Curriculumul STEMKIT refăcut, utilizând principiile de proiectare a instrucțiunilor, poate fi găsit în cadrul portalului de învățare sub formă de obiecte de învățare interactive și va fi livrat cursanților care vor putea urmări cursul interactiv în timpul și ritmul propriu, în timp ce pot utiliza instrumentele de învățare socială oferite pentru a interacționa cu colegii lor (interacționați cu comentarii, forumuri, chat). De asemenea, vor fi furnizate funcții suplimentare de colaborare, cum ar fi notificări, fluxuri de grup, bloguri, partajare de fișiere, întrebări / răspunsuri, vot.

Output O2A1 „STEMKIT CURRICULUM DESIGN & DEVELOPMENT” conține următoarele 5 elemente și anume: Introducere în Scratch 2.0 [P6-ARC], Scratch GPIO (Pini de control GPIO / intrări de recepție) [P3-DANMAR], Introducere în Raspberry Pi Edition Minecraft [P4-HESO / P5-SCHOLE], programare Raspberry Pi GPIO folosind Python [P2-AKNOW] și calcul fizic [P1-ECAM]

Scratch are un număr mic de comenzi, permite schimbul de sprite fără a sparge dependențe, încurajând colaborarea și partajarea codului. Sistemul este întotdeauna activ, fără comutator de rulare / editare, astfel încât comenzile sau fragmentele de cod pot fi executate cu un clic, iar feedback-ul grafic arată execuția. Variabilele și listele au vizualizări concrete, astfel încât efectul operațiilor de date poate fi văzut imediat. (Maloney, 2010)
Capacitatea de a codifica programe de calculator este o parte importantă a alfabetizării în societatea actuală. Când oamenii învață să codeze în Scratch, învață strategii importante pentru rezolvarea problemelor, proiectarea proiectelor și comunicarea ideilor.

Proiecte Scratch - bazate pe unități de învățare vor include diferite discipline și această nouă perspectivă va permite elevilor să aplice ceea ce învață în situații noi, ducând la o învățare mai profundă; elevii vor fi implicați în activități de proiectare, urmărind interese personale, interacționând prin colaborări creative și reflectând asupra experiențelor utile.

Introducerea în Minecraft Pi a permis următoarele puncte cheie de concluzie.
Dacă ați urmărit această resursă cu Raspberry Pi, vă așteptați să:

- Accesați Minecraft Pi și creați o lume nouă.
- Navigați în jurul Minecraft Pi utilizând comenzile de mișcare de pe tastatură.
- Știți cum să plasați și să distrugeți un bloc și să navigați prin diferite tipuri de blocuri in inventarul din joc.
- Conectați Python la Minecraft Pi.
- Utilizați interfața de programare Python.
- Manipulați blocuri folosind cod și scripturi Python.
- Înțelegeți bine restul funcțiilor Minecraft Pi.
- Faceți Minecraft să interacționeze cu lumea exterioară folosind butoane și LED-uri.
- Importați lumi noi și descoperiți pachete de resurse compatibile cu Minecraft Pi.
- Programarea Raspberry Pi utilizând Python a permis următoarele puncte cheie de concluzie.
- După ce ați citit acest capitol, vă așteptați să fi învățat, înțeles și practicat următoarele elemente de învățare:
- Ce este un pin Raspberry Pi GPIO și funcționalitatea sa de bază.
- Cum să realizați circuite de bază cu LED-uri, buzzere, senzori și cum să le conectați la pinii GPIO Raspberry Pi.
- Cum se configurează, se controlează și se programează pinii GPIO utilizând limbajul Python.
- Familiarizarea cu și utilizarea principalelor construcții și sintaxei limbajului Python, inclusiv funcții, variabile, pentru bucle, în timp ce bucle, dacă instrucțiuni.Navigate around Minecraft Pi using the movement controls on your keyboard.

Calculul fizic este o activitate complexă care solicită cursanților să fie conștienți de problemele software și hard în același timp. La nivel introductiv (școala primară), elevii pot lucra cu jucării programabile sau cărămizi programabile și medii de programare drag & drop pentru a învăța fundamentele gândirii algoritmice. Kituri de construcție care au senzori și actuatori preasamblați pentru a fi conectați la un microcontroler fie direct, fie cu un ecran permite copiilor mai mari să ajungă foarte repede la realizări vizibile și tangibile. Odată cu progresul în calcul fizic, atât pe partea hard, cât și pe cea software, elevii sunt supuși unor procese de învățare care întăresc gândirea computațională și competențele cheie care sunt necesare pentru toate aspectele vieții. Cercetările viitoare pe această temă vor investiga, prin urmare, modul în care elevii din diferite grupe de vârstă experimentează activități fizice de calcul și procesele specifice de învățare.