Spencer coding - first steps

levads

<u>Uzstādīšana</u>

Paldies ka atbalsti CircuitMess un esi sveicināts Spencer programmēšanas apmācībā.

Mēs izmantosim CircuitBlocks lai programmētu tikko salikto balss asistentu. CircuitBlocks ir mūsu radīta programmēšanas aplikācija ko esam izstrādājuši. Spencer būs jāprogrammē grafiskajā blokshēmu programmēšanas vidē, kas jums palīdzēs iemācīties pirmos datorprogrammēšanas soļus.

Uzstādīšana

CircuitBlocks pašlaik darbojas un tādām platformām kā- Windows, Linux un Mac OS.

Ja jums ir Windows dators

- 1. Dodies uz https://circuitmess.com/circuitblocks/download/
- 2. Lejupielādē jaunāko versiju priekš Windows
- 3. Nospied uz faila "CircuitBlocks"]
- 4. CircuitBlocks automātiski tiks ieinstalēts

		Ma	nage Cir	cuitBlocks						- 0	×
File Home	Share	View App	Tools								^ 🕜
Pin to Quick Copy access	Paste	从 Cut ₩ Copy path Paste shortcut	Move Cop to • to •	Delete Rename	New iter New folder	Properties	Open • Edit	Select all Select none Invert selection			
Clip	pboard	DC . D . (D)		organise	New	Open		Select			
← → * ↑	> In:	s PC > Data (D:)	> Downloads	> CircuitBlocks					~ O	Search CircuitBlocks	۵
OneDrive	^	Name		C	ate modified	Туре	Size				
This PC		LircuitBlocks	s Setup 1.0.0	2	2.10.2019. 10:22	Application	89.635	KB			
3D Objects											
Desktop											
Documents				Circuit	llocks Setup	_	ПХ				
🕹 Downloads							Land Contraction				
Music				ا ف	nstalling, please wait						
Pictures											
Videos											
Ler (C:)											
Data (D:)				L				_			
🔮 CD Drive (E:)											
Google Drive	File										
Network											
	~										
1 item 1 item sele	ected 87	7,5 MB									=

Ja jums ir Mac dators

- 1. Dodies uz https://www.circuitmess.com/circuitblocks/download
- 2. Lejupielādē jaunāko CircuitBlocks priekš Mac OS
- 3. Pārvieto failu uz "Applications" mapi
- 4. CircuitBlocks tiks instalēts automātiski.



Ja jums ir Linux dators

Ir 2 veidi kā instalēt CircuitBlocks uz Linux

"Vienkāršā" instalācija:

- 1. Dodies uz https://www.circuitmess.com/circuitblocks/download
- 2. Nospied "Linux 64-bit" lejupielādes pogu
- 3. Dubultklikšķis uz faila lai sāktu instalāciju (Ubuntu) Vai

Atver termināli un ieraksti -i <path to the downloaded file .deb> (citām linux versijām)

Applmage instalācija:

- 1. Dodies uz https://www.circuitmess.com/circuitblocks/download
- 2. Nospied "Linux AppImage" lejupielādes pogu
- 3. Nospied uz faila labo taustiņu un izvēlies "Properties"
- 4. Dodies uz "Permissions" un atzīmē "Allow executing file as program"
- 5. Dubultklikšis uz instalācijas un tā sāksies automātiski

	CircuitBlocks-1.0.0.AppImage Properties	
Basic Embler	ns Permissions Open With Notes	
Owner:	robbie - Robert	
Access:	Read and write 👻	
Group	robhie -	
Group.		
Access:	Read and write	
Others		
Access:	Read-only 👻	
Execute:	✓ Allow executing file as program	
SELinux context:	unknown	
Last changed:	Tue 22 Oct 2019 11:53:50 AM CEST	
Usla	Class	_
Неір	Close	e

Ja saskaraties ar kādu problēmu kas saistīta ar instalāciju, lūdzu sazinaties ar mums **contact@circuitmess.com** un atsūtiet ekrānšāviņu un informāciju par savu problēmu.

<u>ūsPamati</u>

Lietotāja interfeiss



Kad atvēriesiet CircuitBlocks, redzēsiet logu kas izskatās šādi.

Tas ir pavisam vienkāršs – Lai sāktu jaunu projektu nospiediet "New project" pogu.

Saglabātie projekti tiks glabāti tur pat blakus 'New project" pogai.

Ja jūs saskaraties ar kādu problēmu lietojot CircuitBlocks, nospiediet "Send error report" loga apakšā. Jums tiks parādīts kļūdas numurs. Lūdzu sazinaties ar mums un norādiet kļūdas numuru.

Jauna projekta izveide (sketch)

Nospied pogu "New project" Jums jāizvēlas ierīce un programmēšanas veids. Pie Device nospiediet **Spencer**.

Pie Sketch nospiediet **Block**.

Tālāk nospied Create pogu.

CITCUILBIOCKS				= U X
Ö Circuit BLOCKS				
		News	sketch	
		Dev	vice:	
		Ringo	Spencer	
		Sketcl	n type:	
Your Sketches		Code	Block	
		Cre	eate	
+		play animation _ failuor	change I by	
New project				
When button gets press				
if if connecter	set SSID CircuitMess	if wifi is connected		

Tad atvērsies attēlā redzamais logs

CircuitBlocks			- a ×
← ■		Spencer disconnected	Close Code <> Run >
Search Q			
😂 Logic		Code	•
C Loops		1 Sinclude (Arduine.b)	53 ×
🖬 Math		2 #include <circuitos.h> 3 #include <spencer.h></spencer.h></circuitos.h>	
A Text		4 5 void setup() { 6 Secial hegin(115300);	
Variables		7 Serial-println(); 8 Spencer.begin();	
Functions	Arduino run first:	9 10 }	
I 1/0		void loop() { 13 LoopManager::loop();	
	Arduino loop forever:		
🖵 LED matrix			
Playback			
Speech			
Ö Time			
		$\stackrel{\circ}{(\pm)}$	
		Ö	

Loga augšējā daļā atradīsiet rīkjoslu ar dažādām pogām.

Bloku izvēlnes rīks atrodas pašā kreisajā pusē- jūs varat izvēlēties blokus un pavilkt tos programmas izveides logā.

Šajā logā jūs veidosiet programmu no krāsainajiem blokiem.

Labajā ekrāna daļā jūs redzēsiet kodu kas rakstīts C++ tiklīdz veidosiet kodu ar blokiem.

C++ ir viena no populārākajām programmēšanas valodām, bet tā nav vienkārša ja iepriekš neesi saskāries ar to.

Tāpēc mēs esam radijuši CircuitBlocks – tu vari veidot kodu no krāsainajiem blokiem un paralēli redzēt kodu C++. Ka pietiekami daudz zināšanas, tad varēsi kodu veidot uzreiz C++ vidē.

Rīkjosla

		Spencer disconnected e	
Search Q		Code	•
C Loops		1 #include <arduino.h> 2 #include <circuitos.h> 3 #include <spencer.h> 4</spencer.h></circuitos.h></arduino.h>	
A Text	Addition our first	<pre>void setup() { 6 Serial.begin(115200); 7 Serial.println(); 8 Spencer.begin(); </pre>	
Functions	Arduino run mist:	9 10 } 11 void loop() (13 LoopManager::loop();	
 Network LED matrix 	Arduino loop forever:	14 15 16 }	
Playback			
Ŭ Time			
		(+)	
		ŏ	

Šī ir īsa pamācība par to ko nozīmē pogas rīkjoslā.

- Atpakaļ uz galveno izvēlni atgriezies uz sākuma logu bez projekta saglabāšanas.
- 2. **Saglabāt/Saglabāt kā** Saglabā jūsu projektu. Pārliecinies ka ik pa laikam saglabā savu darbu, kā arī pirms CircuitBlocks aizvēršanas.
- 3. **Spencer pieslēguma indikators** Sarkanais punkts kļūst zaļs kad tavs Spencer ir pieslēgts pie datora ar USB kabeli.
- 4. **Bin faila eksportēšana** Šī funkcija nodrošina Bin faila(programmas koda) saglabāšanu jūsu datorā.
- 5. **Serial monitor** šī poga atver logu, kurā redzēsiet visus komunikācijas datus kas notiek starp Spencer un jūsu datoru.
- 6. **Close Code** Ar šo pogu, jūs varat aizvērt vai atvērt logu kurā redzams C++ kods. Jūs to varat aizvērt, lai iegūtu vairāk vietas blokiem.
- 7. **Run** Šī poga pārveidos kodu ko izveidojāt, par Spencer balss asistentam saprotamu kodu un nosūtīs to uz pašu Spencer.

Koda logs



Tā sauktais "Koda logs" sastāv no sekojošajām daļām;

- Primārais koda logs kods kas uzrakstīts ar C++ parādīsies šajā zonā kad sāksiet likt kopā blokus. Pamanīsiet ka dažādas koda daļas iekrāsosies dažādās krāsās. Tas ir paredzēts lai programmētājs vieglāk varētu uztvert dažādas koda daļas.
- 2. **Gaišās/Tumšais režīms** jūs varat pārslēgt pamata un teksta krāsas un baltu vai melnu.
- Expand palielina kodu pa visu ekrānu. Nospied velreiz lai atkal to samazinātu.
- 4. Aizvērt Aizver koda logu. Tāda pati funkcija kā "Close code" pogai rīkjoslā.

Bloku laukums

Search Q	K.	
🈂 Logic		
C Loops	2	
🖬 Math		
A Text		
Variables		
Functions		
I/O		
? Network	3	
🖵 LED matrix	Arduino run first:	
Playback	Arduino loop forever:	
Speech		
Ö Time		
	2	4
		-(
		5 (

Šajā laukumā notiek koda veidošana.

Tas sastāv no sekojošajām daļām:

- 1. Meklēšanas rīks meklē blokus pēc nosaukuma
- 2. **Bloku veidi** Šeit bloki ir sadalīti pēc kategorijas. Katra kategorija ir nodalīta savā krāsā.
- Veidošanas zona Šajā zonā jūs varat veidot programmu no blokiem. Cik vienkārši!
- 4. **Centrēšanas rīks** Ja sanāk aizbīdīt ekrānu un pazaudēt blokus, tad nospiežot šo pogu varat iecentrēt visu kodu ekrāna vidū.
- 5. **Tālummaiņas pogas** Tuviniet vai attāliniet darba virsmu.

Soli pa solim

<u>Pirmā programma</u>

Laiks ķērties klāt pirmās aplikācijas programmēšanai

Pirmkārt, jums jāpieslēdz Spencer pie datora ar USB kabeli.





CircuitBlocks jāuzrāda ka "Spencer connected"



Ja CircuitBlocks neredz jūsu Spencer, lūdzu pārbaudiet vai USB kabelis ir pievienots pareizi un USB ports jūsu datorā strādā.

Ja joprojām dators neredz jūsu Spencer, tad iespējams kaut kas nav kārtībā ar datora draiveriem. Draiveri ir tās mazās programmas kas palīdz datoram komunicēt ar Spencer. Ja par šo tēmu rodas jautājumi, tad sazinaties ar mums – **contact@circuitmess.com.**

Izveidosim programmatūru lai Spencer parāda animāciju uz displeja

Atrodiet sadaļu ar nosaukumu "LED matrix" kreisajā ekrāna pusē.

Šajā bloku grupā atradīsiet dažādas komandas kas liek Spencer uz displeja parādīt dažādus attēlus.



Atrodiet bloku ar nosaukumu "Play animation"



Satveriet bloku ar kursoru un aizbīdiet uz jau esošo bloka daļu, pievienojiet tur kur

rakstīts "Arduino run first"

Mēs vēlamies likt Spencer parādīt animāciju tiklīdz tas tiek ieslēgts.



Nospied mazo bultiņu uz "Play animation"bloka un parādīsies visas animācijas kas ir saglabātas Specner iebūvētajā falsh atmiņā. Pagaidām jaunas animācijas pievienot nav iespējams, taču mēs pie tā strādājam, lai ar jaunu CircuitBlocks atjauninājumu tas būtu iespējams.

Taču jums arī ir iespēja iedarbināt katru atsevišķu pikseli un LED paneļa, un to tēmu apskatīsim vēlāk.

Ak jā, turpinam apmācību. Izvēlies "heart"animāciju.



Lieliski! Mūsu pirmā programma ir gatava !

Šis tikko uztaisītais kods liks Spencer parādīt animāciju uz LED matricas. Tiklīdz Spencer tiks ieslēgts, parādīsies animācija un tiks atskaņota bezgalīgi.

Pirms turpinam, nospiedīsim SAVE pogu, un nosauksim projektu.

← ₿	Spencer connected	Close Code <> Run >
Sarch. Q. C. Logic C. Loops Math A. Text Variables	Arduino run first: play animation heart • Arduino loop forever: • void setup() [• Sertal.begin(115209); 7 Sertal.println();	÷ ¤ ×
Functions Functions VO Network LED matrix	Save sketch X	wn(new SerialFlashFileAdapter("GIF-heart.gif"
 Playback Speech Time 		

Kad tas izdarīts, sameklē sarkano pogu RUN , un nospied to

< 🖬 🔤	imple_1	Spencer connected	
Search Q Search Q Image: Composition of the composition of t	Arduino run first: play animation heart • Arduino loop forever:	Code 1 #include A 2 #include A 3 #include C 3 #include C 4 void setup(6 Serial.ep 9 Spencer.l 10 11 LEDmatrix 12 13 14 } 15 16 void lcop() 17 LoopManag 18 19 20 } (•) (•) (•) (•) (•) (•) (•) (•) (•) (•)	<pre>rduino.b> trcuttos.b> pencer.b>) { gin(15200); intln(;) gegin(); aadSettings(); .startAnimation(new Animation(new SerialFlashFileAdapter("GIF-heart.gif"</pre>

Dzeltena ielādes josla parādīsies ekrāna augšdaļā.

Example_1		Spencer connected	tu 📄 Close Code < > Compiling 🔻
vearch. Q Second. Logic Lops Lops Math A Text Variables Functions Vo Vo Network LED matrix Playback Speech Time	Arduino run first: play animation heart Arduino loop forever: Congiligo sietch:	€	<pre>pinclude <arduino.h> finclude <arduino.h> finclude <grauthos.h> finclude <grauthos.h> finclude <spencer.h> void setup() { Serial.println(); Spencer.loadSettings(); LEDmatrix.startAnimotion(new Animation(new SerialFlashFileAdapter("GIF-heart.gif" } void loop() { LoopManager::loop(); } } </spencer.h></grauthos.h></grauthos.h></arduino.h></arduino.h></pre>

Jūsu programma šobrīd ir sastādīšanas procesā

Koda kompilēšana cilvēkam saprotama koda (piemēram krāsainie bloki ko redzat) pārveidošana par iekārtai saprotamu kodu (vieninieki un nullītes ko saprot Spencer)

Par cik šī ir pirmā reize kad tiek kompilēts programmas kods priekš Spencer, tas var aizņemt pāris minūtes. Tas notiek tāpēc, ka CircuitBlocks ir jāapkopo visas svarīgās koda daļas, kas nepieciešamas jūsu Spencer, un jāsaglabā tas datorā. Pirmo reizi apkopojot, tas tiks saglabāts jūsu datorā, un visas nākamās programmas tiks apkopotas daudz ātrāk (yay!).



Ja jūsu kods tika veiksmīgi apkopots un augšupielādēts, jums vajadzētu redzēt sirds animāciju, kas tiek atskaņota jūsu Spencer.

Vai viss kārtībā? Lieliski, ejam tālāk.

Kaut kas nedarbojas? Lūdzu, nospiediet CircuitBlocks mājas lapā ziņojumu Sūtīt kļūdu, sazinieties ar mums pa e-pastu contact@circuitmess.com un sniedziet mums savu kļūdas ziņojuma ID.

<u>Izmēģināsim mainīgos</u>

Nākamajā solī dodamies mazliet tālāk un izmantojam **dažus mainīgos**.

Datorprogrammēšanā mainīgais ir glabāšanas vieta, kurā ir vērtība. Katram mainīgajam ir noteikts nosaukums. Varat saglabāt un mainīt mainīgā vērtību.

Pirmkārt, izveidosim mainīgo. Atrodiet sadaļu ar nosaukumu "Variables" un nospiediet pogu "Create variable ...".



Jums mainīgajam jāpiešķir nosaukums. Šajā gadījumā nosauksim to par "i".



Tagad mums ir mainīgais. Lieliski!

Kad mēs izveidojam mainīgo, tas nav definēts - tam nav vērtības. Startējot mūsu datorprogrammu, katram mainīgajam ir jāiestata vērtība. Tāpēc jums būs

•

nepieciešams bloks "set variable".



Šo bloku ievietojat "Arduino run first" vietā.



Tagad jums ir jādefinē vērtība, kurai mēs vēlamies iestatīt mainīgo.

Sadaļā "Math" atrodiet šo bloku ar nosaukumu "123". Šis bloks ir skaitlisko vērtību bloks, un, ievadot vajadzīgo skaitlisko vērtību, nometiet to zīmēšanas apgabalā.



Novietojat bloku šeit:



Tagad noklikšķiniet uz bloka un ierakstiet vērtību. Iestatiet bloka skaitlisko vērtību uz 0. To var izdarīt, vienkārši ievadot tastatūras skaitli 0 (nulle).



Tagad, kad mums ir izveidots mainīgais un iestatīts uz nulli, mainīsim mainīgo, kad tiks aktivizēts noteikts notikums.

Mēs mainīsim mainīgo, kad jūs uzsitīsit savu Spensera lielo sarkano pogu.

Par laimi, mums ir īpašs bloks, kas definēts Spenser sarkanās pogas piesišanai, un tas atrodas I / O sadaļā. I / O nozīmē "leeja / izvade". Spenser poga ir tā sauktā ievades ierīce, jo tā iedarbina Spencer datoru uz elektrisko impulsu. Spensera displejs ir izvades ierīces piemērs, jo Spensers sūta signālus, lai parādītu informāciju.

Jums jāatrod šis violetais bloks ar nosaukumu "When button gets pressed".



Novieto bloku programmatūras veidošanas logā



Tagad palielināsim mainīgā vērtību par vienu vienību katru reizi kad tiek piespiesta spencer poga.

To var panākt šādi:



random image

Novieto šo bloku kā redzams attēlā:



Tagad darīsim kaut ko interesantu - mēs izmantosim loģisku bloku IF-THEN-ELSE, lai Spencera sejā parādītu dažādas animācijas atkarībā no mainīgā mainīgā lieluma.



Sadaļā "Logic" atrodiet šo sarkano bloku:

Novietojiet šo bloku violetajā blokā "When button gets pressed block".

Tālāk mums būs nepieciešams salīdzināšanas bloks. Šo bloku parasti izmanto, lai salīdzinātu mainīgā vērtību ar fiksētu vērtību (t.i., redzēsim, vai mainīgais i ir vienāds ar skaitli 1).



Novietojiet dīvainas formas salīdzināšanas bloku loģiskajā blokā IF-THEN-ELSE šādi:



Kreisajā pusē vienādības blokā izvēlaties iepriekš sagatavoto mainīgo i.



Paņemiet skaitlisko vērtību bloku no matemātikas sadaļas un novietojiet to

salīdzināšanas bloka labajā pusē šādi:



lerakstiet blokā skaitli 1.

Tas, ko mēs darīsim tagad, ir likt Spenser rādīt dusmīgas sejas animāciju, ja mainīgais ir vienāds ar 1.

Jums jāievieto šāds bloks "Play animation" un nolaižamajā izvēlnē jāizvēlas "angry" animācija:



Liksim Spenser pārvietoties pa septiņām dažādām animācijām, nospiežot viņa lielo sarkano pogu.

Lai to izdarītu, jums piecas reizes jānoklikšķina uz baltās pluszīmes pogas loģiskā IF-THEN-ELSE bloka apakšējā kreisajā stūrī:



Tagad tam būtu jāizskatās šādi:

 Logic Loops Math Text Variables Functions I/O Network LED matrix Playback Speech Time 	earch	Q,
 Loops Math A Text Variables Functions I/O Network LED matrix Playback Speech Time 	¢ °	Logic
 Math A Text Variables Functions I/O Network LED matrix Playback Speech Time 	C	Loops
 ▲ Text ✓ Variables ✓ Functions ✓ I/O ✓ Network ✓ LED matrix ✓ Playback ✓ Speech ✓ Time 	Ħ	Math
 Variables Functions I/O Network LED matrix Playback Speech Time 	Α	Text
 Functions I/O Network LED matrix Playback Speech Time 		Variables
 I/O Network LED matrix Playback Speech Time 	\$	Functions
 Network LED matrix Playback Speech Time 	▶	I/O
 LED matrix Playback Speech Time 	(î	Network
 ▶ Playback ▲ Speech ● Time 	Ţ	LED matrix
 ✓ Speech ✓ Time 		Playback
Ů Time	Ų	Speech
	Ċ	Time

Ievietosim salīdzināšanas bloku katrā jaunizveidotajā loģikas nozarē.

Lai to izdarītu, varat vienkārši ar peles labo pogu noklikšķināt uz jau esošā salīdzināšanas bloka un noklikšķināt uz "duplicate".



Varat arī nokopēt Play animācijas bloku un ievietot to šādi.

Liksim Spenseram rādīt animāciju ar nosaukumu "clouds", kad mainīgais ir vienāds ar 2.



Es aizpildīšu arī pārējās šādas loģiskās filiāles un piešķiršu dažādām animācijām dažādām mainīgā vērtības:



Ņem vērā - ja mēs vēlamies, lai Spenser apritētu šīs septiņas animācijas, mums ir jāatjauno mainīgā vērtība uz nulli, kad tā pārsniedz 6.

Mēs to varam izdarīt, vienkārši pievienojot šos divus blokus loģiskā bloka ELSE atzaram:



Šie divi bloki atiestatīs mainīgā vērtību uz nulli un atskaņos sākotnējo animāciju, ar kuru programma sākās ("heart" animācija).

Ja esat visu pareizi ievērojis, koda beigās vajadzētu izskatīties šādi:



Neaizmirstiet ietaupīt visu savu smago darbu!



Tagad nospiediet lielo sarkano pogu "Palaist" un pārbaudiet kodu pats.

Nu ir laiks apskatīt ko vel mēs varam izdarīt ar Spencer.

<u>Ahoy skaņas efekti.</u>

Spensers var arī sarunāties un radīt smieklīgus trokšņus. Tam ir iepriekš ielādēti dažādi skaņu efekti un dialogi.

Jūs varat augšupielādēt arī savus balss dialogus, bet vispirms atskaņosim dažus iepriekš ielādētos skaņas paraugus.

Sadaļā ar nosaukumu "Playback" ir viss, kas saistīts ar skaņu. Sāksim ar bloka "Play audio sample" vilkšanu un nomešanu zīmēšanas laukumā.



Ievietojiet bloku "Arduino run first" filiālē tā, lai skaņas efekts tiktu atskaņots, tiklīdz sākas jūsu programma.



Tālāk mums jāizvēlas skaņas efekts, kas tiks atskaņots.

Ir vesela sadaļa dažādu paraugu, kurus varat izvēlēties un izpētīt.

Es ņemšu parauga bloku ar nosaukumu "Special".

Search Q	Playback		
State Logic	Arduind	un first:	When button gets pressed -
C Loops	Control	💌 to 🛛	change i ▼ by 1
	play audio sample play a	imation heart 🕶	if $i = 1$ do
Math	play a	dio sample	play animation angry 🔻
A Text	stop playing audio	oop forever:	else if (i • = • 2 do 💬
Variables	when current sample is done		play animation clouds 💌
Functions			else if $i \bullet = \bullet 3$ do Θ
		L	play animation laugh 🔻
	Samples		else if 🖉 i 🔻 = 🔹 👍 do 💬
TNetwork	error mess 🔻		play animation listen 🔻
🖵 LED matrix	generic and -		else if (i 🔹 = 🔹 5) do 💬
🕨 Playback			play animation music 🔻
🍨 Speech	level low -		else if (i 🔹 = 🔹 6 do Θ
Ö Time	special startup 🔻		play animation night 💌
C 1	weather the temperature is \bullet		else Θ
	month January -		set i 🕶 to 0
			play animation heart 💌
	weekday Monday -		•
	number 0 🗸		

Šis bloka paraugs jāievieto blokā "Play audio sample". Nospiežot nolaižamo izvēlni, varat izvēlēties dažādus audio paraugus, kas atrodas audio paraugu kategorijā "Special".

Search Q	
😂 Logic	Arduino run first: When button gets pressed -
<i>a</i>	set i v to 0 change i v by 1
C Loops	play animation heart 🗸 if i v = v 1 do
🖬 Math	play audio sample special random noise 0 - play animation angry -
A Text	Arduino loop forever:
Variables	play animation clouds -
Functions	else if $i \bullet = \bullet 3$ do \bigcirc
I/O	play animation laugh -
? Network	
LED matrix	else if i v = v 5 do \bigcirc
Playback	play animation music -
Speech	else if i • = • 6 do 👄
Ö Time	play animation night 🝷
	else $igodot$
	set i 🔻 to 0
	play animation heart 👻

Ja palaidīsit šo kodu un augšupielādēsiet to savā Spencer, tas atskaņos izvēlēto skaņas efektu, tiklīdz sāksies jūsu programma.

Ja mēs vēlamies pievienot nedaudz vairāk mūsu kodam, mēs varam likt Spenseram atskaņot citu audio paraugu katru reizi, kad nospiežat tā pogu. Jūsu kodam vajadzētu izskatīties šādi, ja vēlaties to darīt:



Atcerieties saglabāt kodu!

Tas bija jautri. Apskatīsim, ko vēl mēs varam darīt.

Vairāk kodēšanas

<u>Laiks likt Spencer runāt</u>

Tagad izmēģināsim kaut ko modernāku - mēs liksim Spenseram ģenerēt un "pateikt" unikālu dialoga katru reizi, kad uzsitīsim viņa lielo sarkano pogu.

Sāciet no jauna, izveidojot jaunu projektu. Saglabājiet to uzreiz un nosauciet to par kaut ko interesantu (es savu esmu nosaucis par "best_code_ever").



Tālāk mums jāsavieno Spenser ar WiFi tīklu.

Spenser ir nepieciešama piekļuve internetam, lai "runātu". Tas darbojas tā, ka Spenser izveido savienojumu ar mūsu serveri un nosūta tekstu, ko vēlas pateikt. Pēc tam serveris ģenerē audio klipu, izmantojot TTS algoritmu, un nosūta to atpakaļ uz jūsu Spencer.

Lai izveidotu savienojumu ar WiFi tīklu, jums būs nepieciešams bloks "Set SSID and password", kas atrodas kategorijā "Network".



Velciet bloku zīmēšanas apgabalā un ievadiet sava WiFi tīkla nosaukumu un paroli.

SSID ir jūsu tīklo	a nosaukums.
--------------------	--------------

arch Q
Cogic
C Loops
Math
A Text
a Variables
Functions
I/O
Network
LED matrix
Playback
Speech
Time

Kad mēs iestatīsim tīkla pieteikšanās datus, liksim Spenser parādīt animāciju "WiFi". Mēs varam to izdarīt, pievienojot šo bloku šeit:

Search Q	
🌣 Logic	
C Loops	Arduino run first
🖬 Math	Arduno fun first.
A Text	set SSID (" MyWifi ") password (" securePassword1234 ")
➡ Variables	play animation wifi 🔻
Functions	Arduino loop forever:
I/O	
중 Network	
LED matrix	
Playback	
🖢 Speech	
Ö Time	

Tagad jums ir jāvelk un nometiet bloku "Connect to WiFi network". Šis bloks sāk savienojumu.



Novieto bloku šeit:

Search	Q	
😂 Logic		
C Loops		
🖬 Math		Arduino run first:
A Text		set SSID " MyWifi " password " securePassword1234 "
Variable	!5	play animation wifi -
Function	ns	
► I/O		connect to WiFi network
? Networ	k	and then
🖵 LED ma	trix	Arduino loop forever:
Playbac	k	
Speech		
Ö Time		

Tagad mums būs nepieciešams loģiskais bloks IF-THEN-ELSE. Mēs izmantosim šo bloku, lai pārbaudītu, vai WiFi tīkla savienojums ir veiksmīgi izveidots.



Sadaļā "network" izmantojiet nosacījumu "wifi is conected".



Novietojiet to loģiskajā blokā IF-THEN-ELSE kā šādu nosacījumu:



Spēlēsim noteiktu skaņas paraugu un animāciju, ja wifi savienojums ir veiksmīgi izveidots.

Esmu izvēlējies smaidošo animāciju un skaņas efektu ar nosaukumu "badum 0", jo tas izklausījās smieklīgi:



Ja WiFi interneta savienojums nav veiksmīgi izveidots, atskaņosim animācijas un audio paraugu ar nosaukumu "no WiFi".



Šī programma darbosies, ja Spensers veiksmīgi izveido savienojumu ar tīkliem. Ja nē, Spensers pateiks, ka tai ir problēmas ar savienojumu ar WiFi tīkliem.

Tagad pievienosim notikumu bloku "When button gets pressed ":

Search Q	
Cogic Logic	
C Loops	Arduino run first: When button gets pressed
🖬 Math	set SSID "MyWifi" password "securePassword1234"
A Text	play animation wifi ▼ connect to WiFi network
Variables	and then if wifi is connected do
Functions	play audio sample special badum 0 🔻
I/O	play animation smile ▼
Network	else
LED matrix	play animation noWifi -
Playback	
Speech	Arduino loop forever:
Ö Time	

Katru reizi, kad tiek nospiesta poga, pārbaudiet, vai WiFi ir izveidots savienojums. Dariet to ar to pašu loģisko bloku IF-THEN un kā nosacījumu ievietojiet "wifi is connected":

 Loops Math Text Variables Functions I/O Network LED matrix Playback Speech Time 	Q Logic	arch
 Math A Text Variables Functions I/O Network LED matrix Playback Speech Time 	Loops	C Loop
A Text	Math	n Math
 Variables Functions I/O Network LED matrix Playback Speech Time 	Text	A Text
 Functions I/O Network LED matrix Playback Speech Time 	Variables	Varia
 I/O Network LED matrix Playback Speech Arduino loop forever: 	Functions	🗘 Func
 Network LED matrix Playback Speech Arduino loop forever: 	I/O	> 1/0
 LED matrix Playback Speech Arduino loop forever: 	Network	? Netw
 Playback Speech Arduino loop forever: 	LED matrix	LED r
Speech Arduino loop forever:	Playback	Playb
Ö Time	Speech	Spee
	Time	Ö Time

Spenseram būs vajadzīgs īss laiks, lai izveidotu vēlamo audio paraugu. Spēlēsim ielādes animāciju, kamēr paraugu ģenerē:

Search Q		
Cogic Logic		
C Loops		
🖬 Math		
A Text	Arduino run first:	
▲ Variables	set SSID " MyWifi " password " securePassword1234" if wifi is connected do	
Functions	play animation wifi • play animation loading0 •)
I/O	and then if wifi is connected do	
Network	play audio sample special badum 0 🗸	
🖵 LED matrix	play animation smile 👻	
Playback	else 🖂	
Speech	play audio sample error no WiFi 🔹	
Ö Time	•	
	Arduno toop forever:	

Lai veiktu nākamo darbību, atveriet kategoriju ar nosaukumu "Speech" un atrodiet bloku "synthesize and say".

Search Q	🔮 Speech	
😂 Logic	Speech to text	
C Loops		
🖬 Math	start listening when done	
A Text	and start processing Arduino run first:	When button gets proceed =
🕰 Variables	when speech gets processed ssip ("MyWifi") passward (" securePassword1234 "	if wifi is connected do
Functions	if error happens	play animation loading0 👻
I/O	if wifi is connected do	
중 Network	Text to speech play audio sam le special badum 0 -	
🖵 LED matrix	synthesize and say "Hello! My name is Spencer:	
▶ Playback	when synthesized	
🞐 Speech	when done talking	
Ö Time	n error happens	
	Arduino loop forever:	

Novietojiet bloku šeit un ierakstiet teikumu, kuru vēlaties pateikt Spenseram. Esmu ierakstījis "Hellow world. This is my first program".

Speech play animation noWifi	Search Q Image: Construction of the search of the sear	Arduino run first: set SSID "CircuitMess 2" password "MAKERphone!" play animation wifi • connect to WiFi network and then if wifi is connected do play audio sample special badum 0 • play audio sample special badum 0 • play audio sample crror no WiFi • play audio sample crror no WiFi • play audio noWifi •	When button gets pressed • if wifi is connected do play animation loading0 • synthesize and say "Hello world. This is my first program" when synthesized when done talking if error happens •
 Speech ine Time 	 Playback Speech Time 	ck play audio sample error no WiFi ▼ play animation noWifi ▼ €	

Kad dialoga balss paraugs tiek ģenerēts (sintezēts), tas tiks automātiski atskaņots.

Kad tas notiks, padarīsim Spensera seju tādu, kā viņš runā, spēlējot animāciju ar nosaukumu "talk".

Kad Spensers pabeidz sarunu, es spēlēšu animāciju ar nosaukumu "wink".

Search Q		
😂 Logic		
C Loops		
🖬 Math		
A Text	Arduino run first:	When button gets pressed -
Variables	set SSID [•] CircuitMess 2 [•] password [•] MAKERphone! [•]	if wifi is connected do
Functions	play animation wifi	play animation loading0 👻
▶ 1/0	and then if wifi is connected do	synthesize and say <u>Heno</u> world. This is my met progra when synthesized play animation talk
? Network	play audio sample special badum 0 👻	
LED matrix	play animation smile •	when done talking play animation wink •
Playback	else	if error happens
Speech	play animation noWifi +	
Ö Time		
	Arduino loop forever:	

Neaizmirstiet saglabāt savu smago darbu, nospiežot pogu Saglabāt augšējā kreisajā stūrī.

Palaidiet kodu un augšupielādējiet to savā Spencer. Katru reizi, nospiežot Spensera pogu, tam vajadzētu skaļi nolasīt tekstu, kuru esat ierakstījis blokā "Synthisize and say".

Ja jūsu Spencer ir problēmas ar savienojuma izveidi ar Wi-Fi tīklu, pārbaudiet tīkla nosaukumu un paroli un mēģiniet izslēgt un ieslēgt spencer.

> Atceries, Spencer strādā tikai ar 2.4GHz (IEEE 802.11 b/g/h) WiFi signālu. 5GHz (IEEE 802.11 ac) tīkli, diemžēl, nav atbalstīti.

<u>Tagad liksim Spencer klausīties un</u> <u>atkārtot!</u>

Lai iegūtu vislielāko kodēšanas izaicinājumu, liksim Spenseram klausīties mūsu teikto un atkārtot šo pašu teikumu ar savu smieklīgo robotu balsi.

Mēs varam vienkārši turpināt kodu, kuru esat izveidojis iepriekšējā nodaļā.

Pirmkārt, izdzēsiet bloku "synthesize and say", ar peles labo pogu noklikšķinot uz tā un nospiežot dzēšanas opciju.



Tagad mums vajadzēs "Start listening" bloku.

Search Q	Speech
Cogic s	inech to text
C Loops	
🖬 Math	start listening when done
A Text	Arduino run first
➡ Variables	when speech gets processed SID ("CircuitMess 2") password ("MAKERphone!") If wifi is connected do
Functions	if error happens play animation loading0 -
I/O	and then if wifi is connected a do
♠ Network	ext to speech play audio sample special badum 0 🔹
🖵 LED matrix	synthesize and say "Hello! My name is Spencer!" smile -
Playback	when synthesized
🞐 Speech	when done talking
Ö Time	
	Arduino loop forever

Novieto to šeit:

SearchQImage: Component of the component of t	Arduino run first: set SSID °CircuitMess 2° password °MAKERphonel* play animation wifi * play animation wifi * play animation wifi * play animation smile * play audio sample special badum 0 * play audio sample error no WiFi * play audio sample error no WiFi *
SpeechTime	Arduino loop forever:

Izmantojot šo kodu, katru reizi, kad nospiežat tā sarkano pogu, Spenser pārbaudīs, vai ir izveidots savienojums ar WiFi, atskaņos ielādes animāciju un sāks klausīties jūsu teikto.

Kad Spensers ir pabeidzis klausīšanos, atskaņosim animāciju, lai mēs zinātu, ka tā tagad apstrādā mūsu balss komandu.



Tagad atrodiet notikumu bloku "when speech gets processed". Šis bloks tiks aktivizēts, kad Spensers apstrādās jūsu balss komandu un saņems stenogrammu no servera.



Turi un velc uz koda laukumu:

Search Q		
🍄 Logic		
C Loops	Arduino run first:	When button gets pressed -
Math	play animation wifi -	if wifi is connected do
A Text	connect to WiFi network	play animation loading0 ▼ start listening
Variables	and then if wifi is connected do	when done play animation heart -
Functions	play audio sample special badum 0 👻	and start processing
I/O	else Θ	
Network	play audio sample error no WiFi 👻	
LED matrix	play animation noWifi -	when speech gets processed
Playback		if error happens
🖢 Speech	Arduino loop forever:	intentResult->transcript intentResult->intent
Ö Time		

Tā kā mēs vēlamies, lai Spensers atkārto to, ko esat viņam teicis, atrodiet bloku "Synthesize and say":



Novietojiet to šeit, jo mēs vēlamies, lai Spensers atkārtotu jūsu vārdus, tiklīdz jūsu balss ir apstrādāta:

Search Q		
🕫 Logic	Andreine wer finde	
C Loops	est SSID ("CircuitMees 2") parameted ("MAKEDphonel")	When button gets pressed -
🖬 Math	play animation wifi *	if wifi is connected do
A Text	connect to WiFI network	play animation loading0 → start listening
☑ Variables	and then if wifi is connected do	when done play animation heart +
Functions	play audio sample special badum 0 -	and start processing
I/O	else 💬	
Network	play audio sample error no WiFi 👻	when speech gets processed
LED matrix	play animation noWifi -	synthesize and say "Hello! My name is Spencer!"
Playback		when synthesized
🖢 Speech	Arduino loop forever:	when done talking
Č Time		if error happens
		if error happens
		intentResult->transcript intentResult->intent

Mēs vēlamies, lai Spensers saka tieši to, ko jūs teicāt. Tāpēc jums jāņem sarkanais vērtību bloks "intentResult-> transcript" un jānomet tas blokā "Synthesize and say".

SearchQSearchLogicImage: Composition of the search o	Arduino run first: set SSID CircuitMess 2" password MAKERphone!" play animation wifi • connect to WiFi network and then if wifi is connected do play audio sample special badum 0 • play animation smile • else © play animation noWiFi • play animation noWifi •	When button gets pressed • if wiff is connected do play animation loading0 • start listening when done play animation heart • and start processing The speech gets processed synthesize and say intentResult->transcript when done ta*ing when done ta*ing if error h.ppers
SpeechTime	Arduino loop forever:	when done tatting if error hopper- if error happen- intentResult->transcript intentResult->intent

Visbeidzot, atskaņosim runājošu animāciju, kad Spensera balss paraugs tiks ģenerēts, un liks viņam pamirkšķināt, kad viņš būs beidzis runāt.

urch Q		
Logic		
C Loops	Arduino run first:	When button gets pressed -
n Math	set SSID " CircuitMess 2 " password " MAKERphone! "	if wifi is connected do
A = .	play animation wifi 💌	play animation loading0 🗸
1 lext	connect to WiFi network	start listening
Variables	if wifi is connected do	when done play animation heart -
Functions	play audio sample special badum 0 👻	and start processing
	play animation smile 👻	
1/0	else 🕞	
Network	play audio sample error no WiFi 💌	
LED matrix	play animation noWifi 💌	synthesize and say intentResult-stranscrin
Playback		when synthesized play animation talk
	Arduino loop forever:	
Speech		when done talking play animation wink
Time		
		if error happens
		if error happens
		intentResult->transcript intentResult->intent

Lieliski, jūsu kods ir gatavs.

Saglabājiet to un palaidiet to savā Spencer.

Piesit Spencer galvu un pasaki viņam kaut ko (t.i., "Sveiks Spenser, tu esi mans draugs"). Spensers apstrādās jūsu balss komandu un atkārtos to ar savu robotu.

Atjaunot Spencer versiju.

<u>Atjaunot Spencer versiju.</u>

Kad esat pabeidzis kodēšanu un vienkārši vēlaties, lai jūsu Spensers atkal būtu "normāls", jums jāatjauno viņa bāzes programmaparatūra.

Tas ir pavisam vienkārši, vienkārši pievienojiet savu Spencer datora USB portam un augšējā labajā stūrī nospiediet pogu "Restore firmware".



Jums tiks piedāvāts logs, kurā jums jāizvēlas ierīce, kurai atjaunojat programmaparatūru.

Protams, izvēlieties Spenseru.



Pagaidiet dažas sekundes, un jūsu Spensers atkal darbosies kā parasti.

Jums tas jādara ikreiz, kad esat pabeidzis kodēt savu Spencer, ja vēlaties, lai viņš atgrieztos pie sākotnējās ārējās funkcionalitātes.

<u>Ko tālāk ?</u>

Jūs esat sasniedzis mūsu pirmās Spencer kodēšanas apmācības beigas, apsveicam!

Es ceru, ka jūs esat tikpat satraukti kā mēs par Spensera nākotni, jo nākamajos programmaparatūras un CircuitBlocks atjauninājumos mēs esam paredzējuši daudz uzlabojumus.

Tikmēr turpiniet patstāvīgi izpētīt un parādiet mums, ko esat paveicis ar savu Spencer, daloties uz CircuitMess kopienas forumu: https://community.circuitmess.com/

Ja jums nepieciešama palīdzība saistībā ar ierīci, kā vienmēr, sazinieties ar mums pa **contacts@circuitmess.com**, un mēs palīdzēsim pēc iespējas ātrāk.

Paldies un turpini gatavot!

