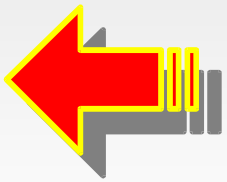
Lab. 10a

Scratch & PicoBoard

Step 1 - Safe-lock Finite State Machine (FSM)



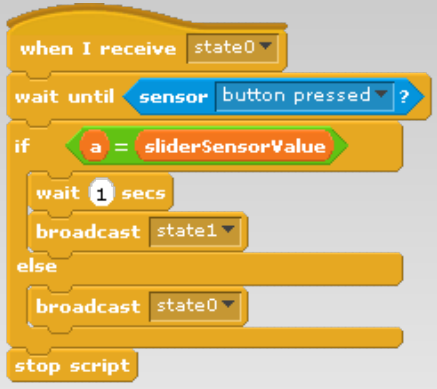
Step 2 - Coin acceptor FSM

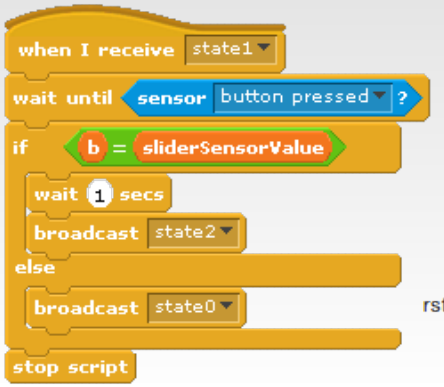
Step 3 - Binary-to-decimal converter

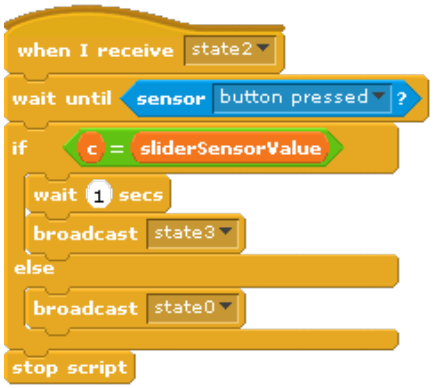
Step 4 - 3x3 numeric touch-pad & homework

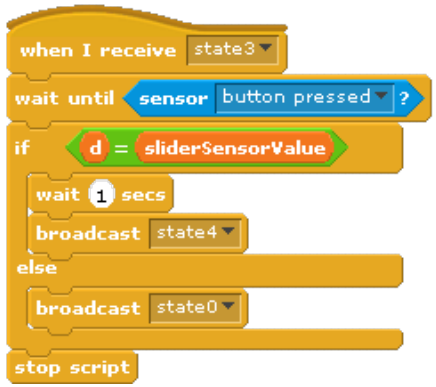
Finite State Machine (FSM)

by a safe lock design example

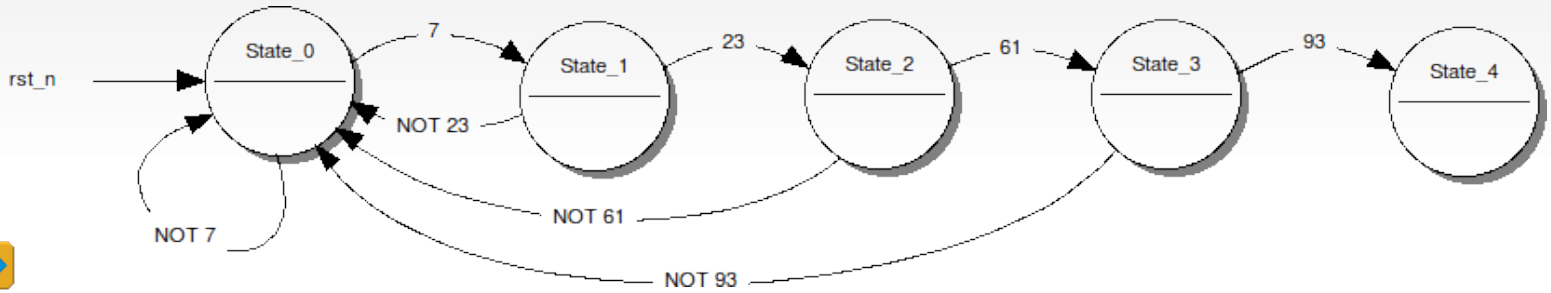








- FSMs are used to model systems which have a limited number of ***states***, ***transitions*** between these states and the ***actions*** taken as a result.
- State transition is a function of ***the current state AND the input*** to the system.



- ***The FSM above:*** number-on-arrow represents what the system senses as an ***input***, thus a ***transition*** from one ***state*** to another can take place. If there is no entry, the system keeps the ***current state***.
- The example represents an acceptor FSM, parsing the secret ***combination*** which, in our example, would un-lock a safe: 7, 23, 61, and 93. In case an unexpected number is entered, the system returns to the first state: State_0. The safe is unlocked, that is, the system is in the final state (State_4), only if the ***correct numbers*** are entered in the ***correct order***.
- On the right hand side, ***a possible FSM implementation*** in Scratch environment is given.

Finite State Machine (FSM)

by a safe lock design example

To Do:

- Change directory to "***Lab10a/Step_1***"
- Open the file "***safeLock.sb***" with Scratch
- Run the code to see it working
  - Use the ***potentiometer*** (or the slider) ***to set values*** also visible on the display
  - Use the ***button to enter*** the selected number to the FSM
- The numbers forming the secret combination can change between 0 and 100. This is not comfortable as the potentiometer (or the slider) is too sensitive to movements (or its gain is too big). Modify the implementation such that the numbers change between ***0 and 10 but not 0 and 100*** anymore.

Lab. 10a
Scratch & PicoBoard

Step 1 - Safe-lock Finite State Machine (FSM)

Step 2 - Coin acceptor FSM

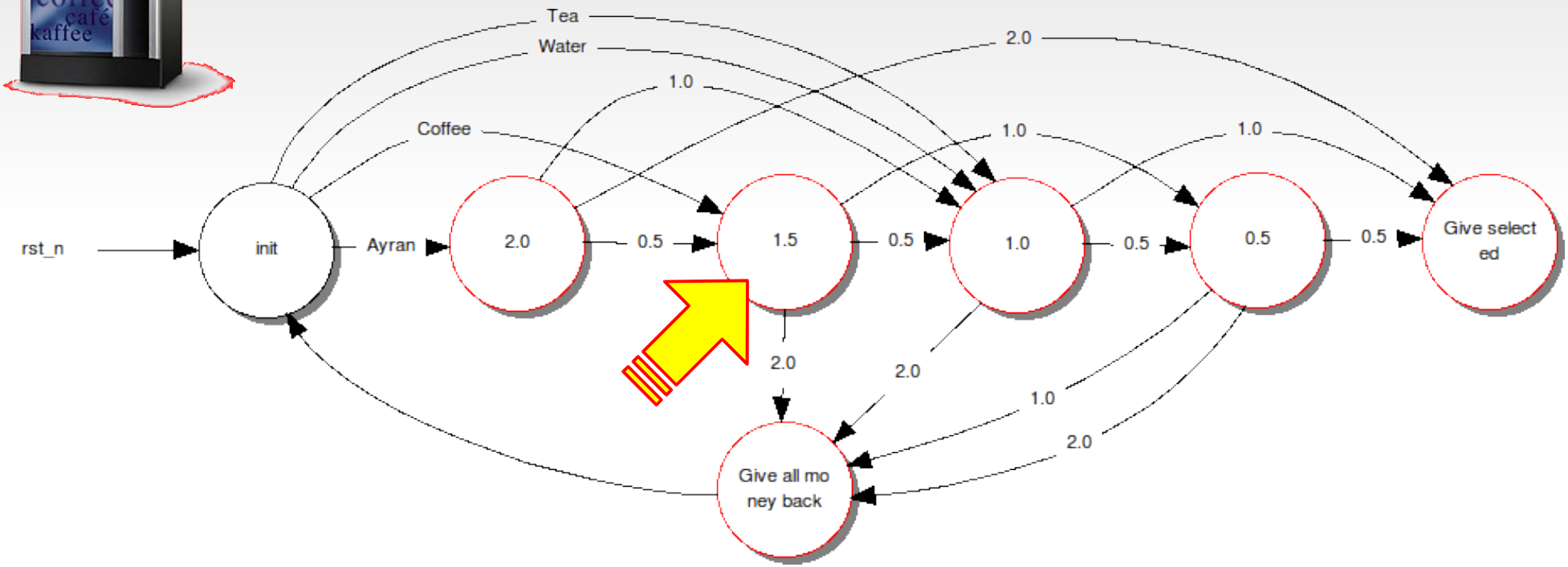
Step 3 - Binary-to-decimal converter

Step 4 - 3x3 numeric touch-pad & homework

Finite State Machine (FSM)

by a coin acceptor design example

- FSMs are used to model systems which have a limited number of ***states***, ***transitions*** between these states and the ***actions*** taken as a result.
- The text on the arrows represent what the system senses as ***input***, thus a ***transition*** from one ***state*** to another can take place. When there is no entry, the system keeps the ***current state***.

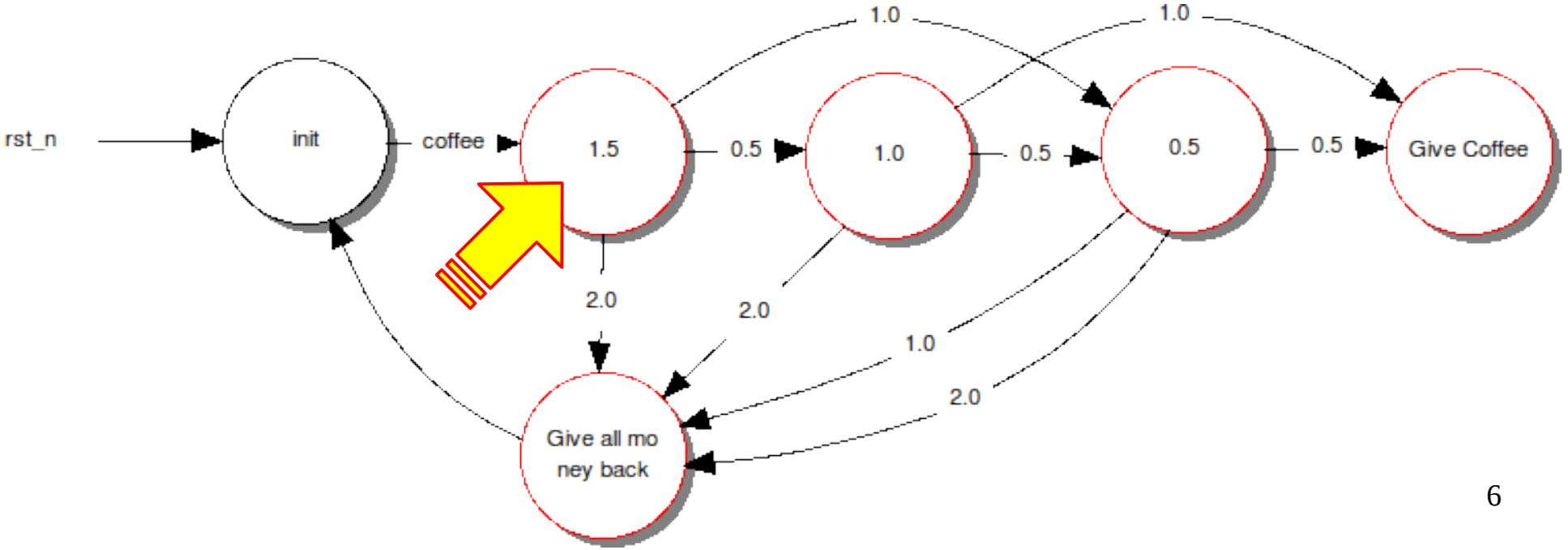


- The example above represents a coin acceptor FSM, expecting the required amount of money for the selected drink where there are three available ***type of coins 2, 1 and 0.5***:
  - ***1.5*** unit for ***caffee***
  - ***2.0*** unit for ***ayran***
  - ***1.0*** unit both for ***water*** and ***tea***

Finite State Machine (FSM)

by a coin acceptor design example – **Coffee case**

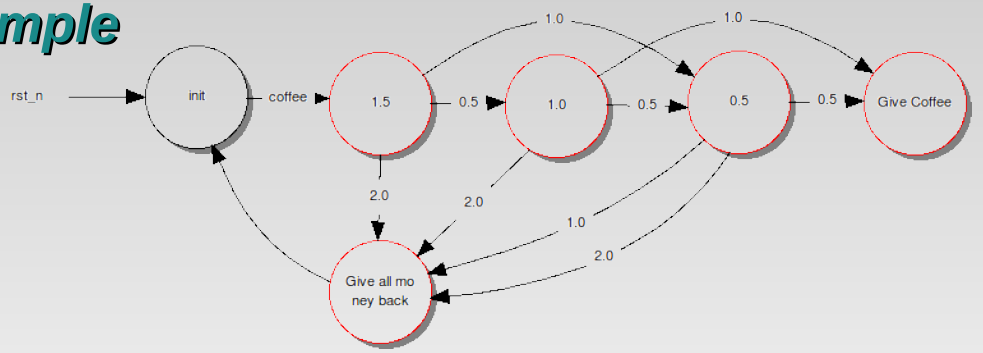
- However this FSM has a limitation; let us consider the ***coffee case***:
  - The machine gives what is selected ***only if*** a client enters the ***exact*** amount; ***otherwise*** all the coins are ***returned***, terminating the operation
  - We would like the machine to have the ***notion*** of "the rest" so that the client is not supposed to enter the exact amount but an amount which can ***exceed what is required***
  - The rest should return to the client
- Implementation is also shown on the next page with annotations.



Finite State Machine (FSM)

by a coin acceptor design example

- ***Signal-slot mechanism*** to communicate or equivalently ***to transition*** from one state to another.



Finite State Machine (FSM)

by a coin acceptor design example

To Do:

- Change directory to "***Lab10a/Step_2***"
- Open the file "***coinAcceptor.fsm***" with Qfsm
- Add/remove states and transitions to the FSM so that it ***returns the rest*** to the user ***and delivers*** what is selected
  - Use ***State > New*** menu item to add a new state
  - Use ***Transition > New*** menu item to add a new transition
  - Use ***Del key to remove*** a selected state and/or a transition
- Open the file "***coinAcceptor.sb***" with Scratch
- Modify the implementation accordingly
  - Use the ***color code to find*** relevant items on the ***left-side bar***

Lab. 10a
Scratch & PicoBoard

Step 1 - Safe-lock Finite State Machine (FSM)

Step 2 - Coin acceptor FSM

Step 3 - Binary-to-decimal converter

Step 4 - 3x3 numeric touch-pad & homework

Binary-to-Decimal Converter

Using ports A, B, C, and D

- We will use the resistance reading ports as 4-bits active-high binary inputs and calculate the decimal correspondence.

$$(abcd)_2 \rightarrow (x)_{10}$$

- The ports A, B, C, and D read unitless resistance values between 0 (***short circuit***) and 100 (***open circuit***). Therefore the ports can function for a given range of resistances connected to them.
- We will use “short cut” or “some resistance” condition as ***logic high*** and “open circuit” condition as ***logic low***.
- Then every time the button is pressed (interpreted as enter), we will calculate their weights in the corresponding decimal value and output the result.
- On the right hand side, a possible implementation in scratch is given.

To Do *(Optional)****:***

- Change directory to “***Lab10a/Step_3***”
- Open “***bin2dec.sb***” with Scratch
- Run the code to see it working

Lab. 10a

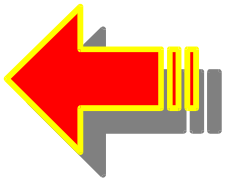
Scratch & PicoBoard

Step 1 - Safe-lock Finite State Machine (FSM)

Step 2 - Coin acceptor FSM

Step 3 - Binary-to-decimal converter

Step 4 - 3x3 numeric touch-pad & homework



3x3 Num Pad Design

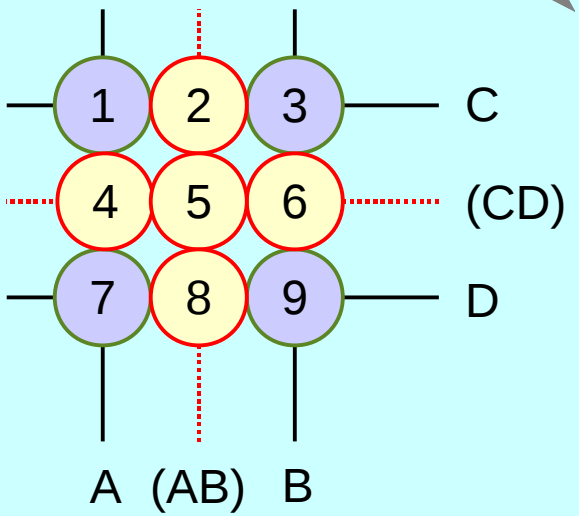
Using ports A, B, C, and D

- We will use the resistance reading ports as 4-bits ***binary inputs*** and ***map*** them onto a 3x3 touch sensitive numeric pad array.
- We will use "*short circuit*" or "*some resistance*" condition as ***logic high*** and "open circuit" condition as ***logic low***. Two possible ***mappings*** are given below:

I. A-B and C-D parallel wire couples cross each other to form a 2x2 array corresponding to the numbers 1, 3, 7, and 9. The other numbers (yellow circles) are mapped as both of the parallel wires are "***fired***".

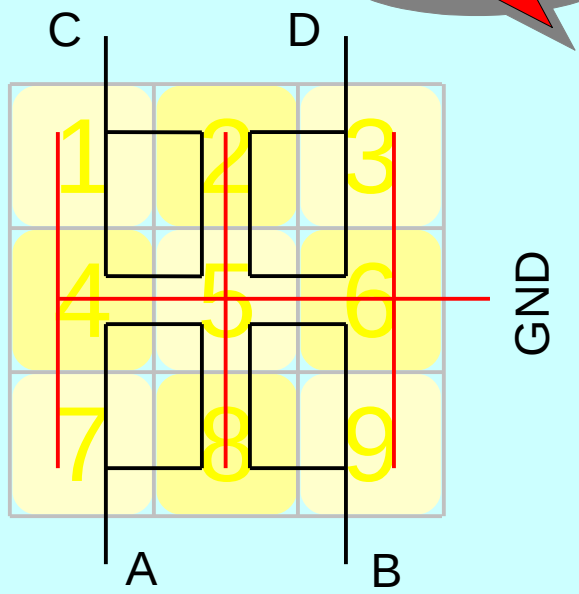
II. A-B and C-D parallel wire couples do not cross each other but instead they are arranged as seen in the figure.

Solution I



Num	Wire
1	A C
2	ABC
3	BC
4	A CD
5	ABCD
6	BCD
7	A D
8	AB D
9	BD

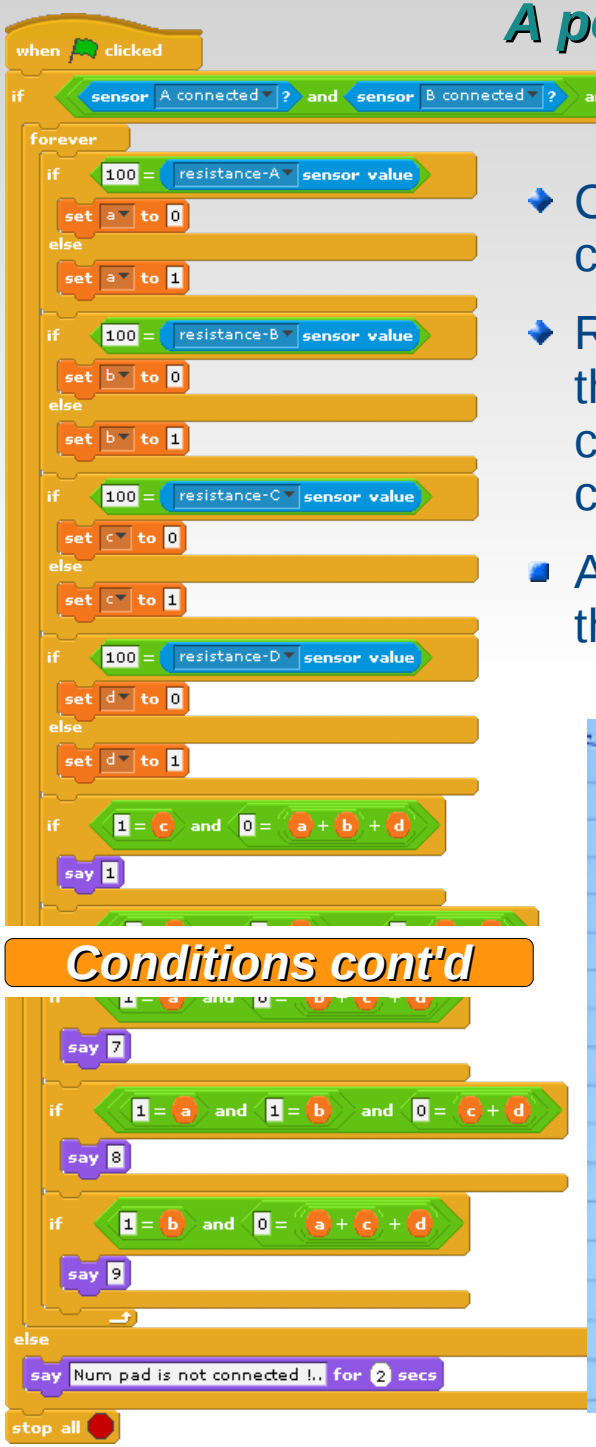
Solution II



Num	Wire
1	C
2	CD
3	D
4	A C
5	ABCD
6	B D
7	A
8	AB
9	B

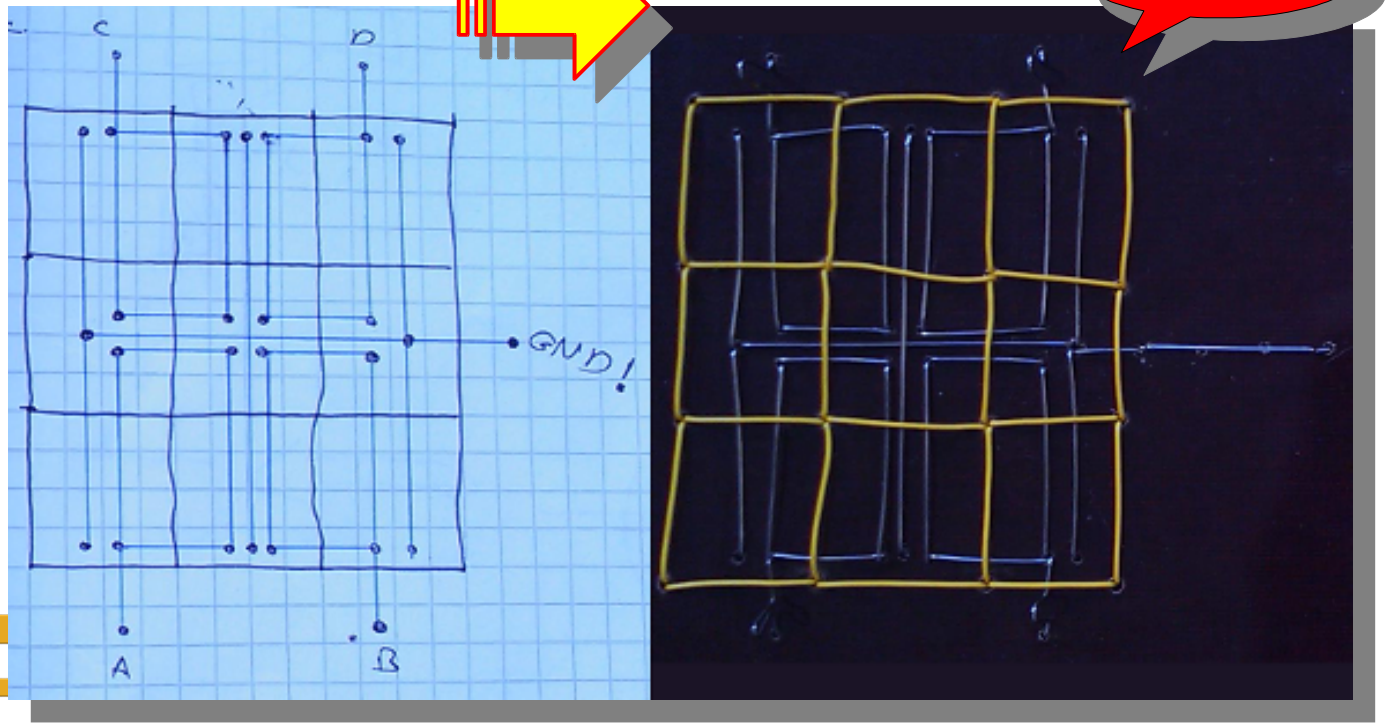
3x3 Num Pad Design

A possible implementation of Solution II



Conditions cont'd

- Check whether the ***ports*** are connected (which in turn connected to the touch sensitive numeric pad
- Read binary values of the wires: if the user is touching one of the numbers, then the resistance between the ***ground*** and the corresponding wires should be less than 100 (i.e. not open circuit).
- Apply the mapping information for the wire values to "***decode***" the action



3x3 Num Pad Design

A possible implementation of Solution II

To Do:

- Change directory to "***Lab10a/Step_4***"
- Open the file "***3x3_Pad.sb***" with Scratch
- Connect the crocodiles to the hand-made numeric pad
  - ***Un-flagged*** crocodiles are the ***Gnd***
- ***Run*** the code to see it working

Lab. 10a
Scratch & PicoBoard

Homework assignment:

- Design a ***4x4*** numeric pad similar to the one just presented
- Design a possible ***wiring geometry*** which can fulfill the requirement and write down the ***mapping*** information
- Implement the ***decoder*** accordingly in Scratch
- Consider ***merging*** a FSM and the numeric touch pad you designed to implement a ***vending machine***

Lab. 10a

Scratch & PicoBoard

Step 1 - Safe-lock Finite State Machine (FSM)

Step 2 - Coin acceptor FSM

Step 3 - Binary-to-decimal converter

Step 4 - 3x3 numeric touch-pad & homework