Boi-Ukeme Joseph

101053347

josephboiukeme@cmail.carleton.ca

remote Home Appliance control model

**Methodologies for Discrete-Event Modelling and Simulation**

**PART I**

This is a model to control home appliances remotely by dialling a number. It gives the home owner the flexibility of being able to turn home appliances on or off, this can have a positive impact as regards safety, energy/power conservation, and time conservation. This model can be extended to various applications where devices, equipment or facilities have to be controlled remotely, for example industrial applications. The remote home appliance control model consists of seven (7) components.

**Model Structure**

Lighting

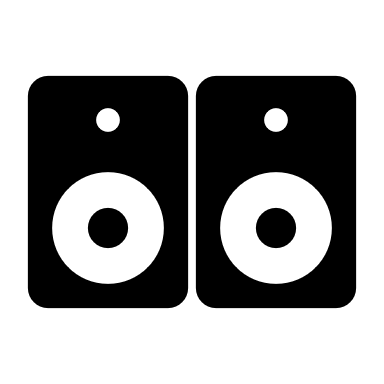
Fridge

Heat

r1Out

r2Out

r3Out



DTMF

MCU

callIn

gsmOut

r3In

dIn

callIn

s

Relay1

Relay3

Relay2

r1In

GSM

m1Out

dOut

mcuIn

scOut

r2In

scIn

scOut

m2Out

m4Out

SoundCh

m3Out

Control

Figure 1.0 Remote Home Appliance Control Model

**The *GSM*** receives a call from the homeowner and sends a signal to the ***DTMF*** corresponding the dialling tone frequency of the key pressed by the home owner, every button on the homeowner’s mobile device (external input) has a unique frequency (this is described in detail in table 1.0). The ***DTMF*** receives the output from the GSM, decodes it, and sends a unique binary code corresponding to a DTMF frequency. The ***MCU*** receives inputs from the DTMF and performs two (2) functions, it sends input to the relay models which will decide which relay to turn ON/OFF and sends status of each relay to the Soundch model. The ***SoundCH*** model simply outputs a pre-recorded sound describing the status of the relays as selected by the MCU. The ***Relays (1,2,3)*** models are simple switches that turn the connected appliances ON or OFF, the state of the relay is equal to the state of the appliance that relay is connected to. The output from the model is the connected to the various appliances (Fridge, Heat and Lighting in this case).

From the figure 1.0 above, the GSM receives a **callIn** input with a specific dial tone, it then checks if the dial tone against a list of allowed tones, if the frequency of the tone is present in the list, the GSM outputs that frequency through its output port **gsmOut** otherwise it discards the input, the DTMF receives a frequency input which corresponds to the output from the gsmOut through its input port **dIn,** then it converts this input frequency to an integer and send this integer through its output port **dOut.** The MCU receives the value of dOut through its input port **mcuIn** and decides which of its four (4) output ports would be activated, the MCU has output ports m1Out m2Out m3Out and m4Out respectively, m1Out m2Out m3Out are used to control Relay1, Relay2, and Relay3 respectively, while output port m4Out is used to control the soundch. The Relays receives integer input from the MCU from their input port rIn and sends out an output of either 0 (to turn OFF) or 1 (to turn ON) the connected appliance.

**Conceptual Model Summary**

1. Frequency Conversion Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Button pressed** | **Frequency** | **Binary output** | **Switching Model** | **Appliance Controlled** | **Soundch Recording** |
| 1 | 10 | 0001 = 1 | Relay1 | Heat ON | 1 |
| 2 | 20 | 0010 = 2 | Relay1 | Heat OFF | 2 |
| 3 | 30 | 0011 = 3 | Relay2 | Fridge ON | 3 |
| 4 | 40 | 0100 = 4 | Relay2 | Fridge OFF | 4 |
| 5 | 50 | 0101 = 5 | Relay3 | Lighting ON | 5 |
| 6 | 60 | 0110 = 6 | Relay3 | Lighting OFF | 6 |
| 7 | 70 | 0111 = 7 | NA | NA | NA |
| 8 | 80 | 1000 = 8 | NA | NA | NA |
| 9 | 90 | 1001 = 9 | NA | NA | NA |
| 0 | 100 | 1010 = 10 | NA | NA | NA |
| \* | 110 | 1011 = 11 | NA | NA | NA |
| # | 120 | 1100 = 12 | NA | NA | NA |

**Table 1.0 Frequency conversion.**

From the table above, we observe that only six (6) buttons are used, hence the model can be extended to control more appliances by including additional buttons.

1. The model is always powered up, hence there is no need to model behaviour of the power supply unit.

**PART II**

The remote home appliance control model (RHACM) consists of components: the sub models GSM, soundch, relays and a coupled model (with two atomic models, DTMF and MCU) as shown in figure 1.0. The RHACM has one (1) input (callIn) and four outputs (scOut, r1Out, r2Out, and r3Out). In summary the RHACM should receive an appropriate callIn input and control one or more devices via it output ports r1Out, r2Out, and r3Out, which gives the status of the device controlled through its output port scOut.

**Formal Specifications**

The formal specifications <S, X, Y, δint, δext, λ, ta> for the atomic models are defined as follows:

**GSM:**

S = {passive, active}

X = {callIn}

Y = {gsmOut}

δint (active) = passive

δext (s,e,x)

{

A = {10, 20, 30, 40, 60}

δext (callin ∈ A, state == passive)

{

received = callIn

state = active

}

δext (callIn ∉ A, state == passive)

{

// do\_nothing

}

δext (callIn, active) = active

λ(active)

{ send *received* to port gsmO*ut* //frequency output

}

ta(passive) = INFINITY

ta(active) = receivingTime

}

**DTMF:**

S = {passive, active}

X = {dIn}

dOut

dIn

dIn

dOut

Y = {dOut}

δint (active) = passive

δext (dIn, passive)

{

decoded = dIn/10 // conversion of frequency to binary

state = active

}

δext (dIn, active) = active

λ(active)

{ send *decoded* to port dO*ut* //frequency output

}

ta(passive) = INFINITY

ta(active) = decodingTime

}

**MCU:**

S = {passive, active}

X = {mcuIn}

Y = {m1Out, m2Out, m3Out, m4Out}

δint (active) = passive

mcuin

mcuIn

m1Out

δext (mcuIn, passive)

m2Out

{

m3Out

controlled = mcuIn

state = active

m4Out

}

δext (mcuIn, active) = active

λ(active)

{ send *controlled* to port m4O*ut*

}

λ(active, controlled ==1| | controlled ==2)

{ send *controlled* to port m1O*ut*

}

λ(active, controlled ==3| | controlled ==4)

{ send *controlled* to port m2O*ut*

}

λ(active, controlled ==5| | controlled ==6)

{ send *controlled* to port m3O*ut*

}

ta(passive) = INFINITY

ta(active) = controlTime

}

**SOUNDCH:**

S = {passive, active }

scOut

scIn

scIn

scOut

X = {scIn}

Y = {scOut}

δint (active) = passive

δext (scIn, passive)

{

record = scIn

state = active

}

δext (scIn, active) = active

λ(active)

{ send *record* to port scO*ut*

}

ta(passive) = INFINITY

ta(active) = controlTime

}

**RELAY1:**

S = {passive, active}

r1Out

r1In

r1In

r1Out

X = {r1In}

Y = {r1Out}

δint (active) = passive

δext (r1In, passive)

{

switched = r1In

state = active

}

δext (r1In, active) = active

λ(active)

if (position is even)

{

send 0 to port r1O*ut*

}

else (send 1 to port r1Out)

ta(passive) = INFINITY

ta(active) = switchingTime

}

**RELAY2:**

r2Out

r2In

r2In

r2Out

S = {passive, active}

X = {r2In}

Y = {r2Out}

δint (active) = passive

δext (r2In, passive)

{

switched = r2In

state = active

}

δext (r2In, active) = active

λ(active)

if position is even

{ send 0 to port r2O*ut*

}

else send 1 to port r2Out

ta(passive) = INFINITY

ta(active) = switchingTime

}

**RELAY3:**

S = {passive, active}

X = {r3In}

r3Out

r3In

r3In

r3Out

Y = {r3Out}

δint (active) = passive

δext (r3In, passive)

{

switched = r3In

state = active

}

δext (r3In, active) = active

λ(active)

if position is even

{ send 0 to port r3O*ut*

}

else send 1 to port r3Out

ta(passive) = INFINITY

ta(active) = switchingTime

}

The formal specifications <X, Y, D, {Mi}, {Ii}, {Zij}, SELECT > for the coupled model Control and Home appliance control model are defined as follows:

**Control**

X = {dIn};

Y = {m1Out m2Out m3Out m4Out };

D = {MCU, DTMF};

I(DTMF) = MCU;

I(MCU) = self;

Z(DTMF) = MCU;

Z(MCU) = self;

SELECT: ({DTMF,MCU}) = DTMF;

**RHCM:**

X = {callIn};

Y = {r1Out, r2Out, r3Out, scOut};

D = {GSM, Control, Relay1, Relay2, Relay3, soundch};

I(GSM) = {Control};

I(Control) = {Relay1, Relay2, Relay3, soundch};

I(Relay1,2,3) = self;

I(Soundch) = self;

Z(Control) = {Relay1, Relay2, Relay3, soundch};

Z(Relay1,2,3) = self;

Z(Soundch) = self;

SELECT: ({GSM, Control, Relay1, Relay2, Relay3, soundch}) = GSM;

({Control, Relay1, Relay2, Relay3, soundch }) = Control;

({Relay1, Relay2, Relay3, soundch}) = Relay1, Relay2, Relay3;

**Test Strategies**

The atomic models and coupled models were tested using the “black box” testing method. Test cases were created by adding different combinations of inputs to the event file (*.ev*), the simulation was then run to check whether the outputs contained in the (.out) files corresponded with the outputs expected from the models (atomic and coupled)

**Part III**

The following test cases were created to test each model.

**Test Cases and Execution Analysis**

**GSM:**

The input of the GSM is an integer number which is a factor of 10 which corresponds to a frequency representing the dial tone of a digit on the cell phone of the home owner (see table 1.0). For this model the input is limited to 6 frequencies (10, 20, 30, 40, 50 and 60) dial tones for (1, 2 3 4 5 and 6) respectively. The GSM simply outputs the received frequency so long as it is within the specified set. The GSM model is designed such that it only receives one input frequency at a time and it takes the *receivingTime (2 time units)* to transmit that frequency hence discards any input that occurs before the time elapses. The *gsm.ev* was created to test the functionality of the gsm model.

00:00:00:00 callIn 10

00:00:11:00 callIn 20

00:00:22:00 callIn 30

00:00:33:00 callIn 40

00:00:44:00 callIn 50

00:01:70:00 callIn 60

00:01:80:00 callIn 70

00:01:90:00 callIn 80

00:01:99:00 callIn 90

From the output file below, we can see that the GSM model does not generate outputs for any input that is not in the set (last 3 events above), this is an expected behavior of the model. To extend this model for more appliances then the set of frequencies can be increased.

00:00:02:000 gsmout 10

00:00:13:000 gsmout 20

00:00:24:000 gsmout 30

00:00:35:000 gsmout 40

00:00:46:000 gsmout 50

00:02:12:000 gsmout 60

**ATOMIC MODEL DTMF:**

The input of the DTMF is a number which corresponds to a frequency representing the output from the GSM model. The DTMF is a simple frequency converter than converts the input frequency to a decimal output. The DTMF model is designed such that it only receives one input frequency at a time and it takes the *decodingTime (1-time unit)* to convert the input frequency and generate a decimal output, hence, discards any input that occurs before the time elapses. The *dtmf.ev* was created to test the functionality of the DTMF model.

00:00:10:00 dIn 10

00:00:20:00 dIn 20

00:00:30:00 dIn 30

00:00:40:00 dIn 40

00:00:50:00 dIn 50

00:01:10:00 dIn 60

00:01:20:00 dIn 70

00:01:30:00 dIn 80

00:01:40:00 dIn 90

00:01:50:00 dIn 100

From the output file below, we can see that the DTMF model converts the input frequencies to decimals as expected.

00:00:11:000 dout 1

00:00:21:000 dout 2

00:00:31:000 dout 3

00:00:41:000 dout 4

00:00:51:000 dout 5

00:01:11:000 dout 6

00:01:21:000 dout 7

00:01:31:000 dout 8

00:01:41:000 dout 9

00:01:51:000 dout 10

**ATOMIC MODEL MCU:**

The input of the MCU is a number which corresponds to a decimal output from the DTMF model. The MCU is a controller that receives an input and sends it to one or more of its output ports depending on the value of the input it receives. The mcuIn should receive an integer input corresponding to converted frequency outputted from the DTMF. The MCU model is designed such that it only receives one input at a time and it takes the *controlTime (5-time units)* to send this input to the desired output port, hence, discards any input that occurs before the time elapses. The *mcu.ev* was created to test the functionality of the MCU model.

00:00:10:00 mcuIn 1

00:00:20:00 mcuIn 2

00:00:30:00 mcuIn 3

00:00:40:00 mcuIn 4

00:00:50:00 mcuIn 5

00:01:00:00 mcuIn 6

00:02:10:00 mcuIn 98

00:03:10:00 mcuIn 68

00:01:30:00 mcuIn 688

From the output file below, we can see that the MCU model sends outputs to various ports as expected. It is important for the MCU to only send outputs when the input is what is expected, hence from the output file we can see that there are no outputs for the events in bold above.

00:00:15:000 m4out 1

00:00:15:000 m1out 1

00:00:25:000 m4out 2

00:00:25:000 m1out 2

00:00:35:000 m4out 3

00:00:35:000 m2out 3

00:00:45:000 m4out 4

00:00:45:000 m2out 4

00:00:55:000 m4out 5

00:00:55:000 m3out 5

00:01:05:000 m4out 6

00:01:05:000 m3out 6

**Soundch**

This model was designed to receive an input from the MCU and outputs an integer number corresponding to a recording. The soundch takes *recordTime (10-time units)* to complete the process. The *soundch.ev* was created as follows to test the soundch model.

00:00:15:00 scIn 1

00:00:30:00 scIn 2

00:00:45:00 scIn 3

00:00:60:00 scIn 4

00:00:90:00 scIn 5

00:01:70:00 scIn 6

The output shows expected results as followed:

00:00:25:000 scout 1

00:00:40:000 scout 2

00:00:55:000 scout 3

00:01:10:000 scout 4

00:01:40:000 scout 5

00:02:20:000 scout 6

**Relay1, Relay2, Relay3**

The input to the relay is usually a positive integer value of between 1 and 6. There should not be an instance where an input arrives at the relay when it is “switching” the previous input. The *relay.ev,* files are used to test the relays. Since all the relays are similar the tests and output for Relay1 is used to illustrate this.

00:00:10:00 r1In 1

00:00:30:00 r1In 2

00:00:45:00 r1In 3

00:00:52:00 r1In 4

00:01:25:00 r1In 5

00:01:35:00 r1In 6

The relay sends out two possible outputs, either a “0” to turn appliance OFF or a “1” to turn appliance ON. The output file shows expected results as follows.

00:00:10:000 r1out 1

00:00:30:000 r1out 0

00:00:45:000 r1out 1

00:00:52:000 r1out 0

00:01:25:000 r1out 1

00:01:35:000 r1out 0

**Coupled Model Control:**

The coupled model Control consists of the DTMF model and the MCU model. The output of the MCU takes the output of the DTMF as its input. The test results obtained is similar to that obtained when testing the MCU. The *control.ev* is created as follows:

00:00:00:00 dIn 10

00:00:11:00 dIn 20

00:00:22:00 dIn 30

00:00:33:00 dIn 40

00:00:44:00 dIn 50

00:01:70:00 dIn 60

The output is deterministic, and we obtained the desired results as follows:

00:00:06:000 m4out 1

00:00:06:000 m1out 1

00:00:17:000 m4out 2

00:00:17:000 m1out 2

00:00:28:000 m4out 3

00:00:28:000 m2out 3

00:00:39:000 m4out 4

00:00:39:000 m2out 4

00:00:50:000 m4out 5

00:00:50:000 m3out 5

00:02:16:000 m4out 6

00:02:16:000 m3out 6

**Coupled Model RHACM:**

The coupled model RHACM is the top model which integrates sub-model GSM with coupled model controller and sub models soundch, Relay1, Relay2 and Relay3. A callIn input which is a frequency with a value which is a factor of 10 is fed into the RHCM and the RHCM turns the corresponding relay ON/OFF through one of its relay outputs, it also sends a record out through its output scOut. The *rhacm.ev* was created to test the model.

00:00:00:00 callIn 10

00:00:11:00 callIn 20

00:00:22:00 callIn 30

00:00:33:00 callIn 40

00:00:44:00 callIn 50

00:01:70:00 callIn 60

00:01:80:00 callIn 70

00:01:90:00 callIn 80

00:01:99:00 callIn 90

The output is deterministic. It can be observed that a relay is turned ON/OFF (output 1 or 0) and then an output recording is sent through scout (1,2,3,4,5). This is the expected output. The results are shown below.

00:00:08:000 r1out 1

00:00:18:000 scout 1

00:00:19:000 r1out 0

00:00:29:000 scout 2

00:00:30:000 r2out 1

00:00:40:000 scout 3

00:00:41:000 r2out 0

00:00:51:000 scout 4

00:00:52:000 r3out 1

00:01:02:000 scout 5

00:02:18:000 r3out 0

00:02:28:000 scout 6