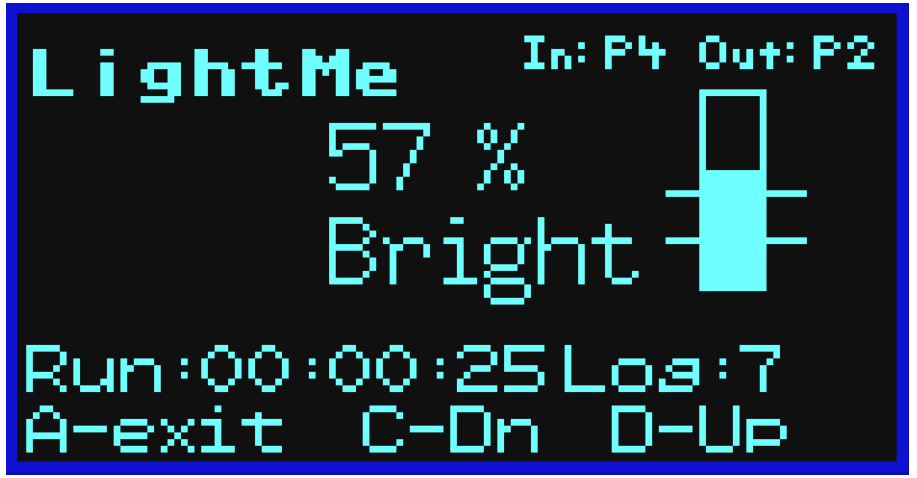
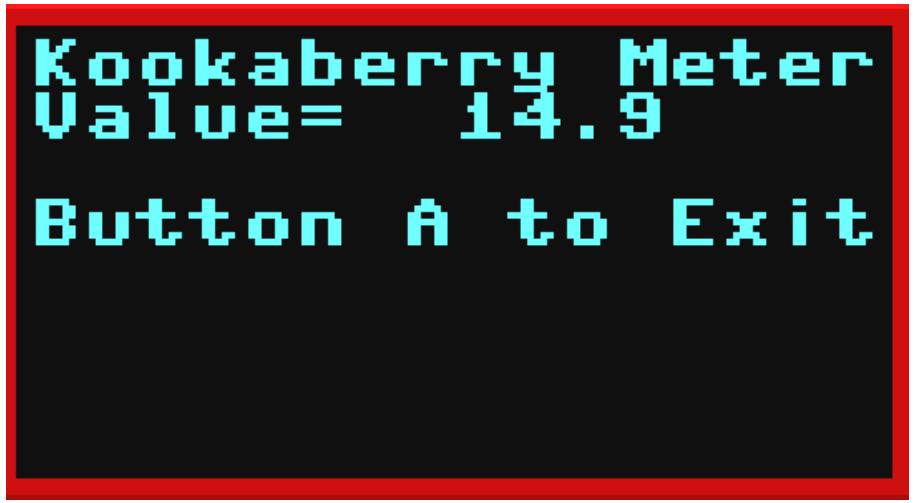
Digital Technologies

Recreating the LightMe App in KookaBlockly

Stages 3&4





kookaberry

About this lesson plan

The STEM Journey Class Kit contains Kookaberries and peripherals that can be connected together and operated as a digital system under the control of pre-loaded MicroPython applications (apps).

Many of these pre-loaded apps can be recreated by students using the KookaBlockly visual programming editor

Recreating the LightMe app

This preloaded app measures the level of ambient light and generates an output signal that can be used to light an LED when it fall below an adjustable threshold.

The full MicroPython code for the preloaded app can be inspected using the KookaIDE text- based editor, but its basic functionality can be recreated using KookaBlockly.

The underlying MicroPython code can be displayed as it is generated in KookaBlockly.

Practical Outcomes

Students will learn to create a customized app that is common in the real world.

Learning Outcomes

Students will use peripherals; position and format text on a screen; create and manipulate strings and variables; receive, transform, and transmit data; scale readings; and implement logic conditions and thresholds.

Prior Knowledge

Familiarity with visual programming editors such as Scratch, and with the following Tutorials

- STEM Journey Class Kit
- Getting Started
- Exploring the Kookaberry
- Introduction to digital systems
- Using the KookaManager

Teacher Resources

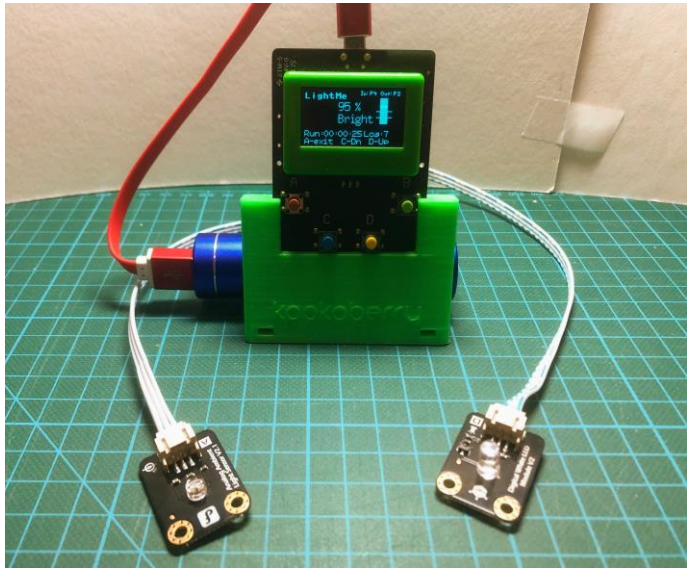
- STEM Journey Class Kit
- Windows PC
- KookaManager (full version).

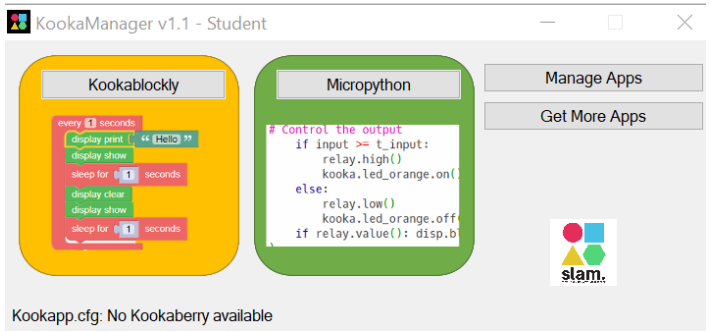
Mapping to curriculum

This is provided in the final slides

Student Resources

- Your resources include
  - A Kookaberry and its power supply
  - Ambient light sensor and LED
  - Connecting leads
  - The Kookaberry website where you can find full details of the LightMe app and its associated peripherals
  - The KookaManager app on your Windows PC





kookaberry

The Kookaberry LightMe App

How it works

The pre-coded Kookaberry LightMe app measures the level of ambient light and sends the data back to the Kookaberry on Pin 4 as an analogue signal..

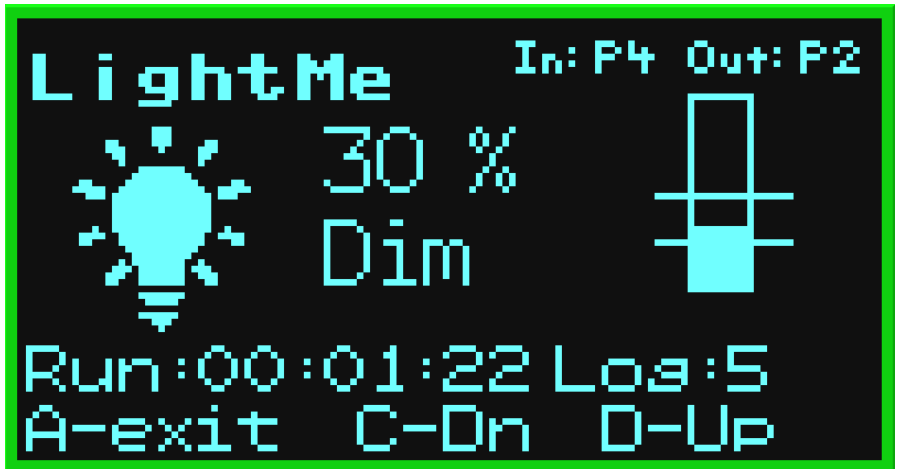
The percentage level of the signal as a proportion of the full voltage available at Pin 4 (Vcc or 3.3 volts) is shown in a vertical histogram (bar) on the screen.

The threshold value of light below which the algorithm will activate the peripheral on P2, is adjustable using Buttons C&D.

The reason for two threshold values is explained in the app description on the website.

The variation in light level over time is also logged and recorded at intervals set in the _config app in a csv file in the Kookaberry's USB memory.

Full details of data logging can be found in the Tutorial: Data Logging on the Kookaberry website



kookaberry

The Kookaberry LightMe App

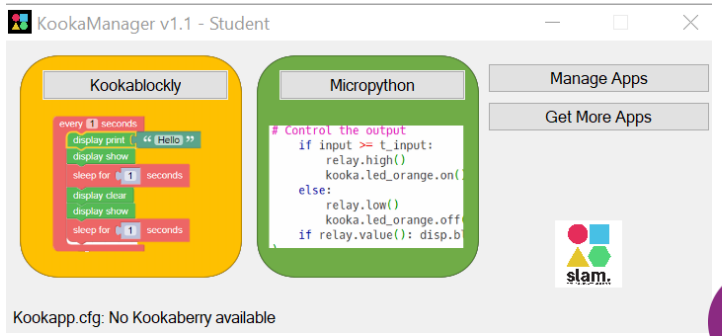
What does its code look like?

- Open KookaManager on your Windows PC and connect a Kookaberry.
- Click on the text “MicroPython” and open the KookaIDE which is an editor for the text-based MicroPython code.
- Find the code for the LightMe app by clicking in the “Scripts” box; selecting “Demos” and then click in the “Choose a script” box to see all the available python files. Click on LightMe.py and the code will appear in the right and window. Scroll down to inspect…..

Professional programmers create python code directly in this type of text-based editor, but they have to be very careful obeying all of its syntax rules and remembering all of its component parts.

It is often simpler to use a much more intuitive visual programme editor like KookaBlockly, which hides the syntax behind easily maneuverable jigsaw blocks on a virtual canvas

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00])

bulb = framebuf.FrameBuffer(bitmap, 29, 32, framebuf.MONO_VLSB)

disp = kooka.display
params = config('Kappconfig.txt')   # read the configuration file
interval = int(params['INTV'])      # use interval from the configuration file
fname = __name__ + '.csv'        # The name of the log file
# set up the datalogger
dlog = logger.Dlog(__name__+'.csv',int(params['INTV']),'Timer,Light %,Threshhold %,Switch %' )    # Create the datalogger instance
dlog.start()    # Start the datalogger

_time_zero = time.ticks_ms()   # initialise the logging duration timer

relay.low() # switch the relay off to begin

while not kooka.button_a.was_pressed():
    timer_run = int(time.ticks_diff(time.ticks_ms(), _time_zero) / 1000) # seconds since beginning
    timestr = '%0.2d:%0.2d:%0.2d'% (int(timer_run/3600),int((timer_run/60)%60), int(timer_run%60))
    disp.fill(0)
    disp.setfont(fonts.mono8x8)
    disp.text('%s' % __name__, 0, 10)
    disp.setfont(fonts.mono5x5)
    disp.text('In:%s Out:%s' % (s_plug, r_plug), 74, 6)
    disp.setfont(fonts.mono6x7)
    disp.text('A-exit', 0, 63)
    disp.text('C-Dn  D-Up', 50, 63)
    disp.text('Run:%s' % timestr, 0, 54)
    disp.text('Log:%d' % dlog.log_counter, 80, 54)
    if relay.value(): disp.blit(bulb, 6, 14)
# Adjust the lighting threshhold using the C and D keys
    if kooka.button_c.was_pressed():
        if t_light >= 15: t_light -= 5    # decrement bright threshhold
        t_dark = int(t_light / 2)
    if kooka.button_d.was_pressed():
        if t_light <= 85: t_light += 5    # increment bright threshhold
```

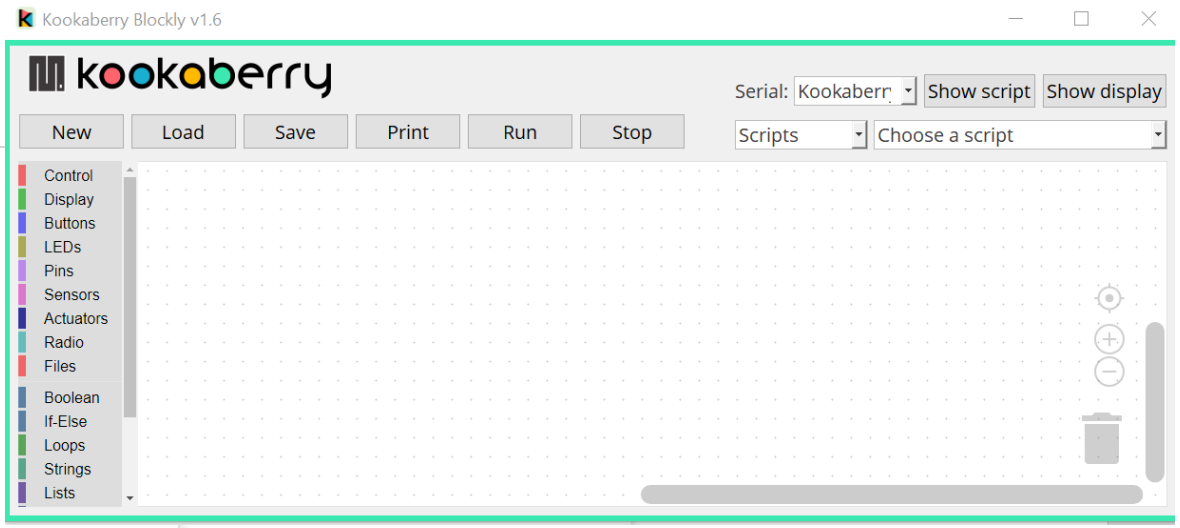


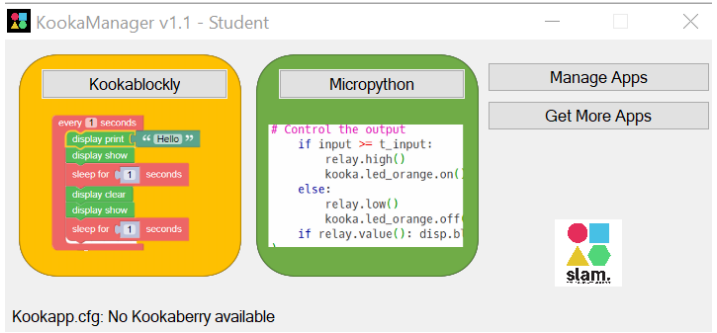
Recreating the LightMe App
Using KookaBlockly

- Open KookaManager (either the teacher or student version) on your Windows PC and connect a Kookaberry.
- Click on the text “KookaBlockly” and open KookaBlockly which is a visual programming editor for the text-based MicroPython code.

This will open the KookaBlockly canvas ready to get started on recreating the preloaded LightMe app.

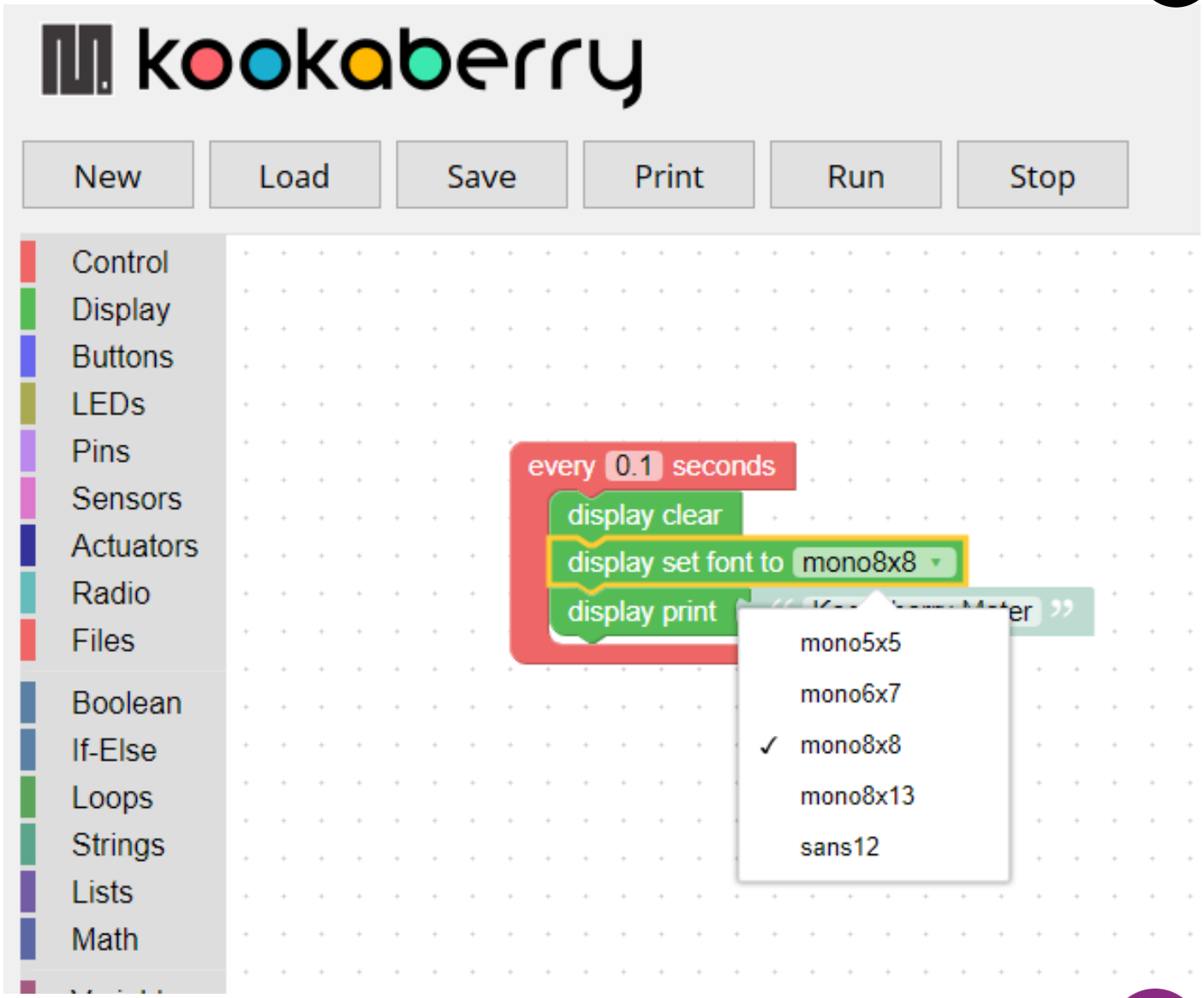
Visual programme editors must be developed for the specific device they are programming. The Makecode editor for instance is customised in different versions for the various devices whose code it is editing (micro:bit, Arduino, Arcade,etc)





Step 1: Setup display title

- Open KookaBlockly on your computer and connect your Kookaberry to it.
- Drag the top Control block onto the canvas and set it to check the instructions inside it every 0.1 seconds.
- Drag “display clear”, “display set font to” and “display print” blocks inside the Control loop.
- Change the “Hello” text to “Kookaberry Meter” and select the mono8x8 font size



kookaberry

Step 2: Create a Variable

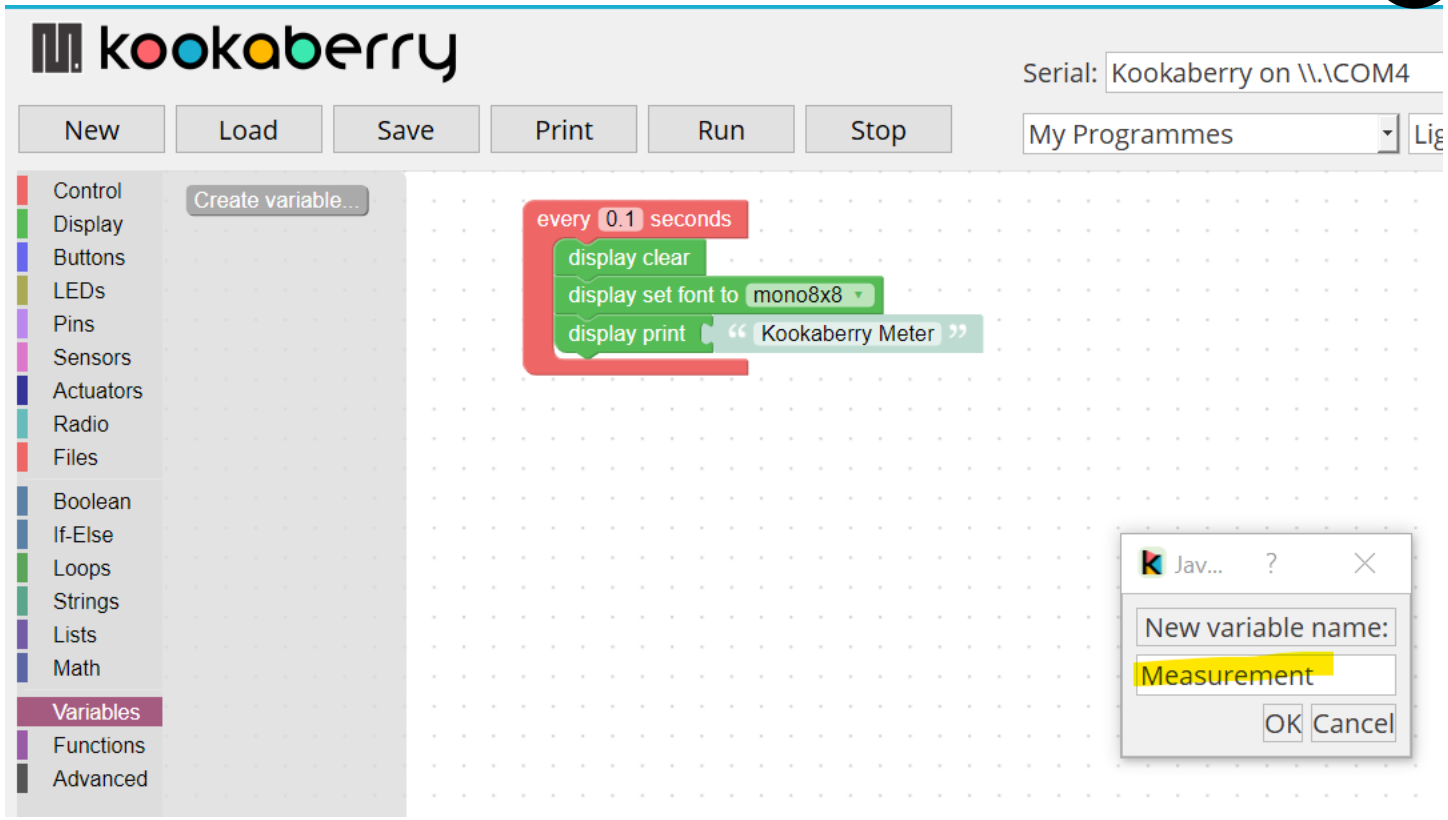
- Create a Variable from the Variables menu.

Clicking on the Create Variable box will display a popup window where you can name the variable.

- Name the variable “Measurement”. Click OK when finished.

This variable is used to measure the voltage coming from the ambient light sensor which will be on Pin 4.

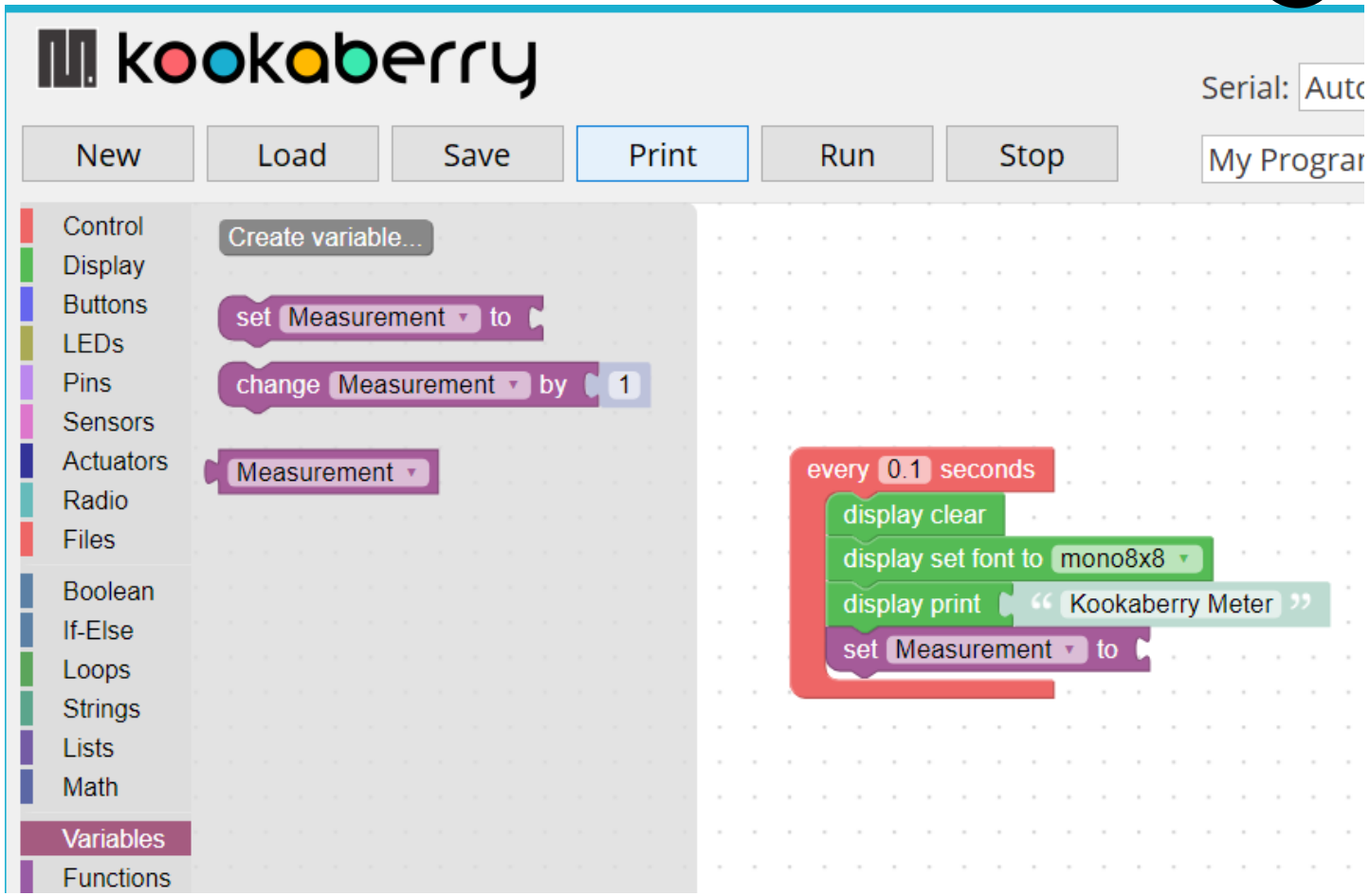
It will appear a few times in the algorithm.



Step 3: Set the Variable

Creating the variable will create three blocks that will allow for setting and changing the variable.

- Drag the “set Measurement to” block inside the Control loop.



kookaberry

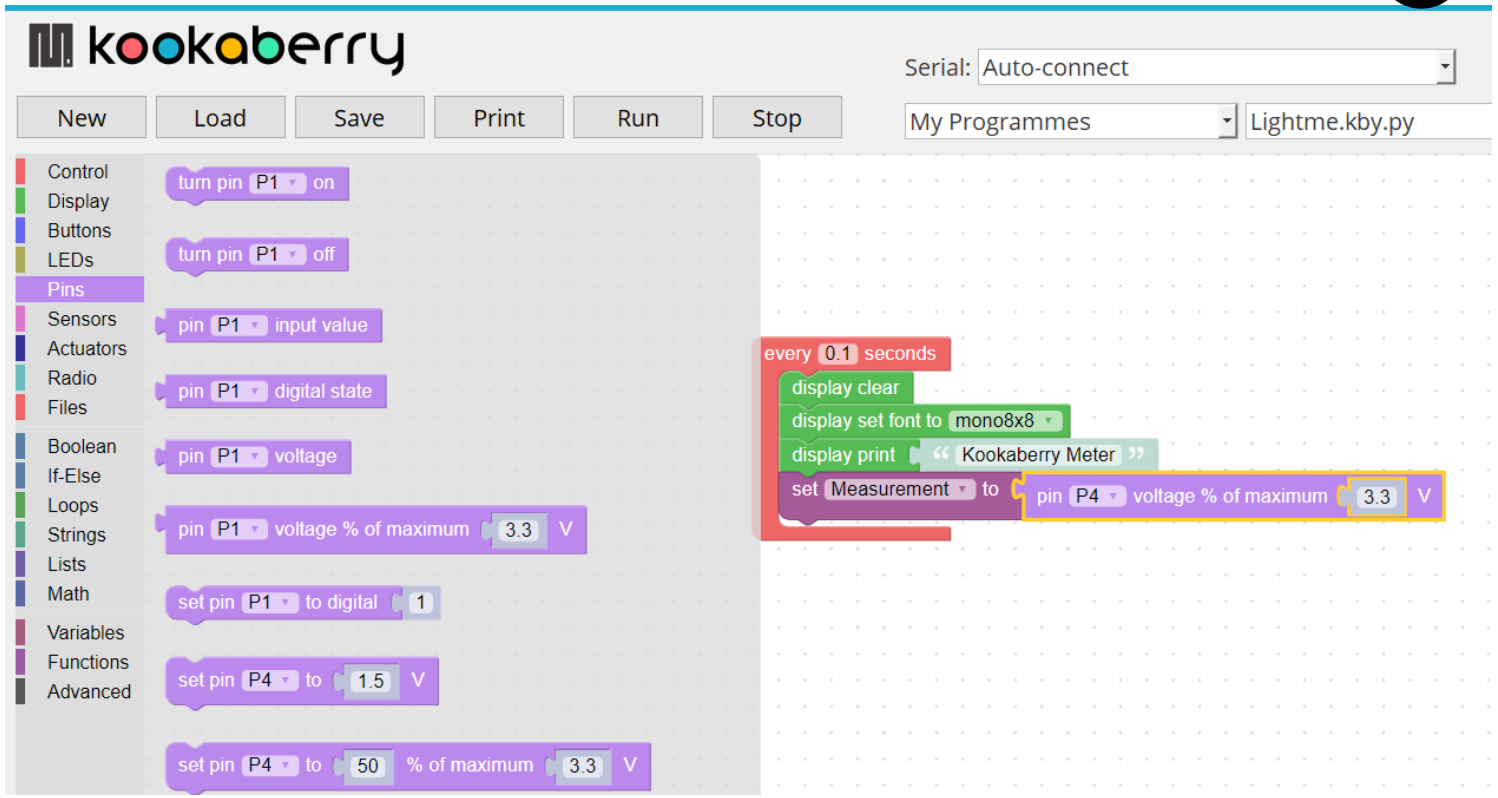
Step 4: Scaling

We must now tell the Kookaberry which pin the sensor is plugged into and make the maximum analogue signal (the ambient light sensor is an analogue device) the same as the maximum voltage (VCC) of the Kookaberry (3.3v).

This is called "scaling" because a sensor with a very small voltage scale (ie 0 to 1v) would only use a third of the scale available to the Kookaberry (0 to 3.3v)

- Drag the scaling block from the Pins menu and set pin to Pin 4

The default maximum is Vcc (3.3v)

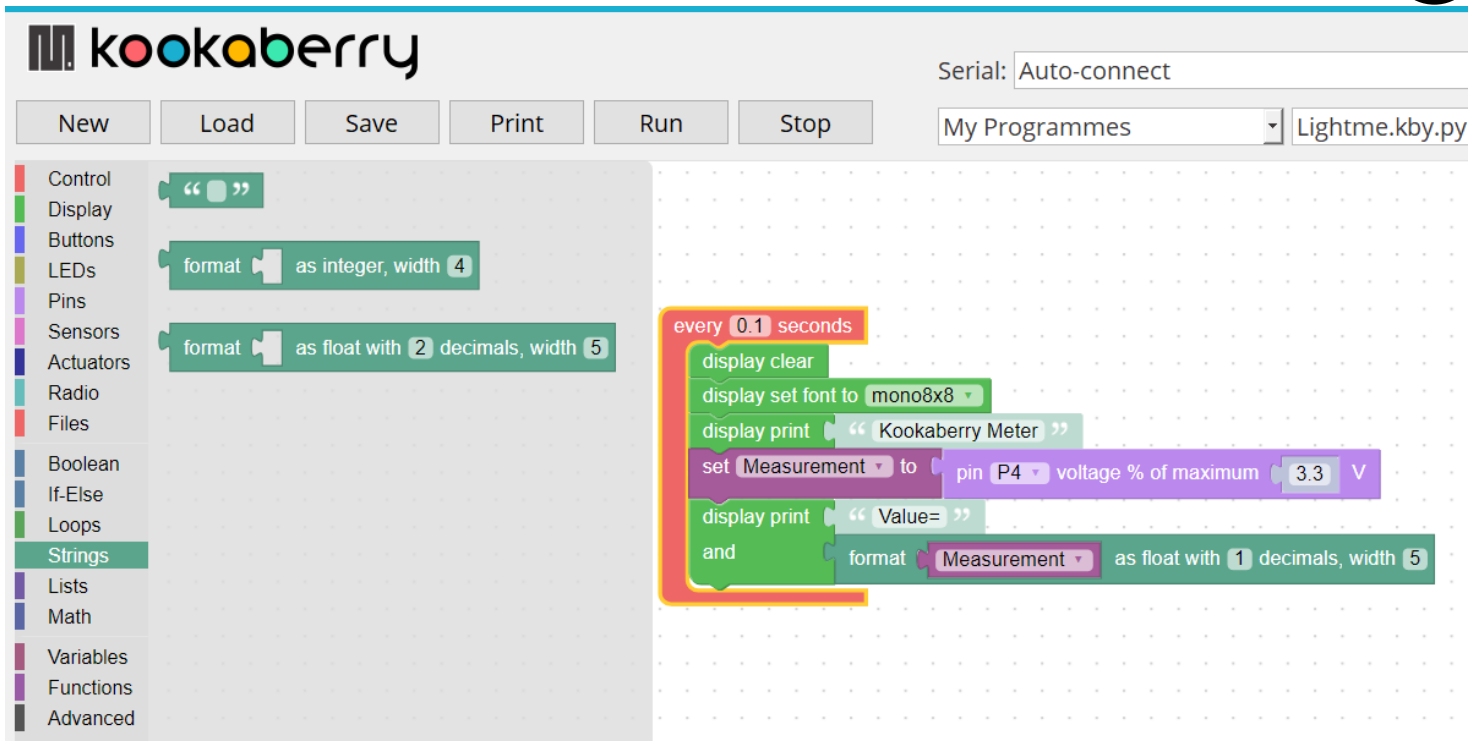


Step 5: Display Reading

We will now display the reading on the screen to one decimal point.

- Drag the “display print “Hello” and “123”” block into the control loop and change the text string to “Value=“
- Replace the “123” math string with the bottom format block from the Strings category. *Drag the block over the maths block…..*
- Drag the Measurement variable into the format space and change the float value to 1 decimal point

The width is measured in pixels. 5 pixels is enough for a space and 4 characters (ie 67.5)



kookaberry

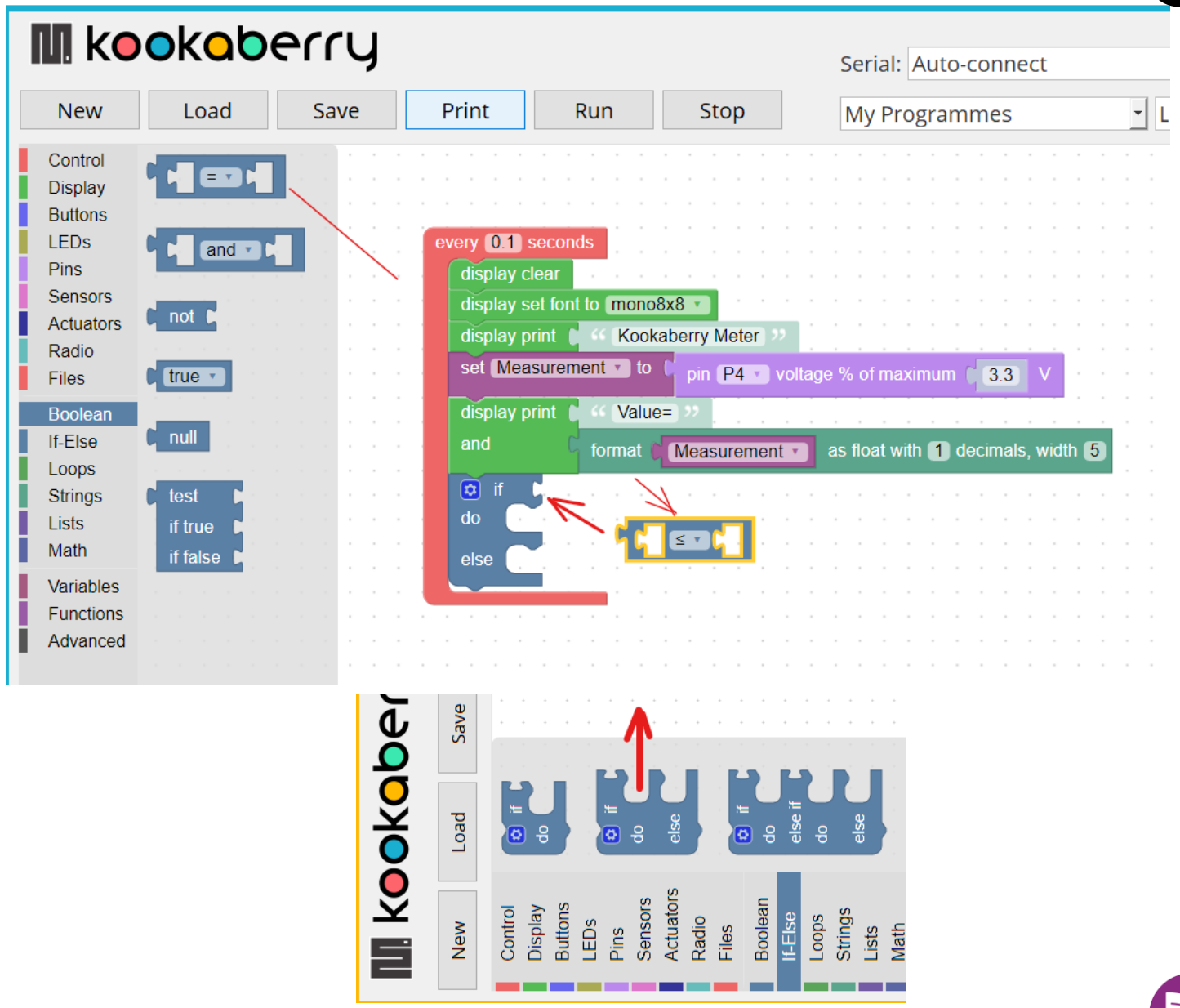
Step 6: Logic

We will now set up the logic conditions which turns a light on when the ambient light falls below a preset level.

- Drag the “if/do/else” block into the loop.
- Drag the top Boolean function block into the canvas and attach it to the “if” statement

The Boolean function compares two values against a set of mathematical conditions. The default condition is “=“.

- Change the condition to “Less than or equal to” using the small dropdown menu in the block



kookaberry

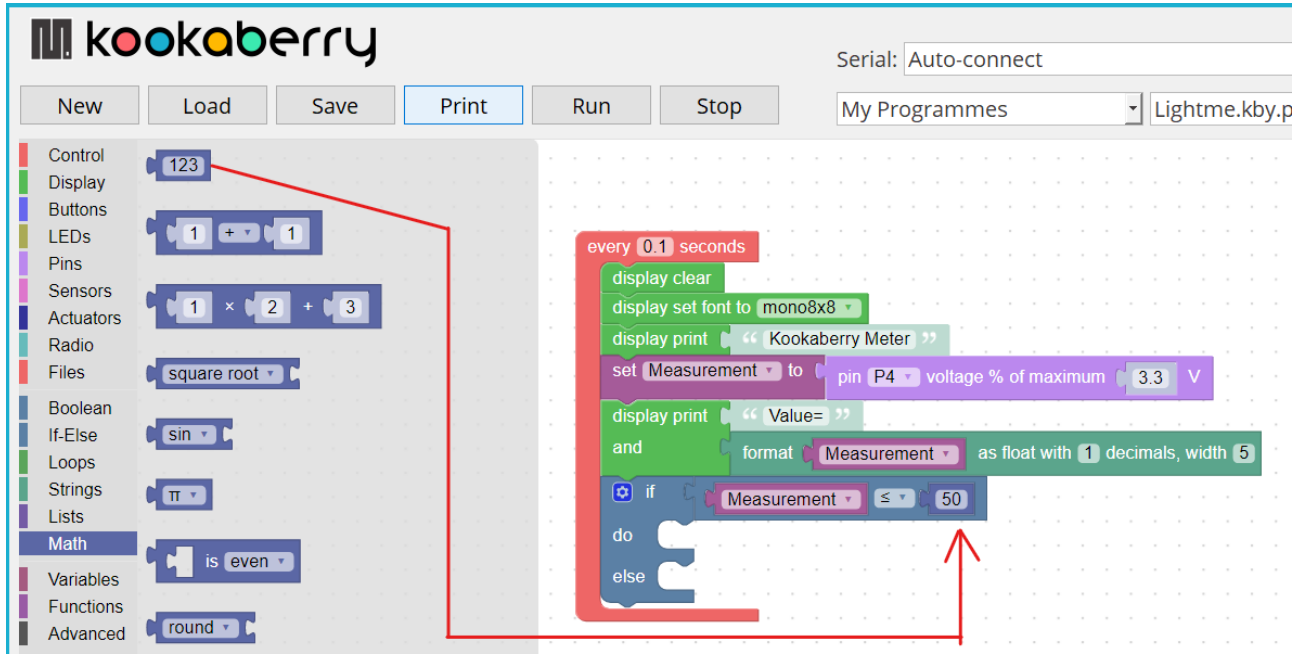
Step 7: Light threshold

- Drag the “Measurement” variable into the first segment of the Boolean comparison block
- Drag the top “123” block in the Maths category into the second segment and change the value to “50”

When the sensor is receiving maximum light, the value on the screen is 100%.

If the value is set to “50”, the light will be switched on when the value falls to 50% or lower.

This is called the threshold value and can be easily changed.



kookaberry

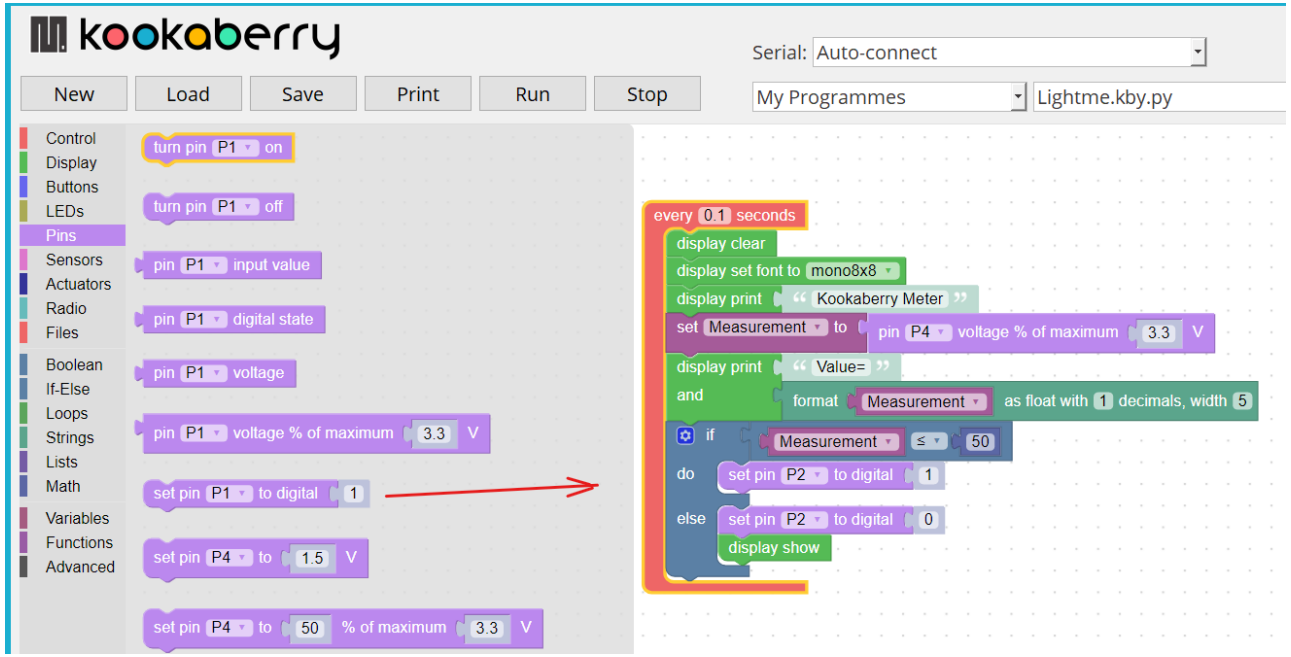
Step 8: Light the LED

- Drag the “set pin [P1] to digital [1]” block into “do” socket of the “if/do/else” block and change the pin to P2.
- Drag another “set pin [P1] to digital [1]” block into “else” socket of the “if/do/else” block and change the pin to P2.
- Drag a “display show” block to the bottom of the control loop

If *the light level is less than 50 (%) then “digital 1” will set Pin to HIGH/ON.*

If *the light level is greater than 50 (%) then “digital 0” will set Pin to LOW/OFF.*

“display show” will display the “display print” instructions on the screen every 0.1 seconds



kookaberry

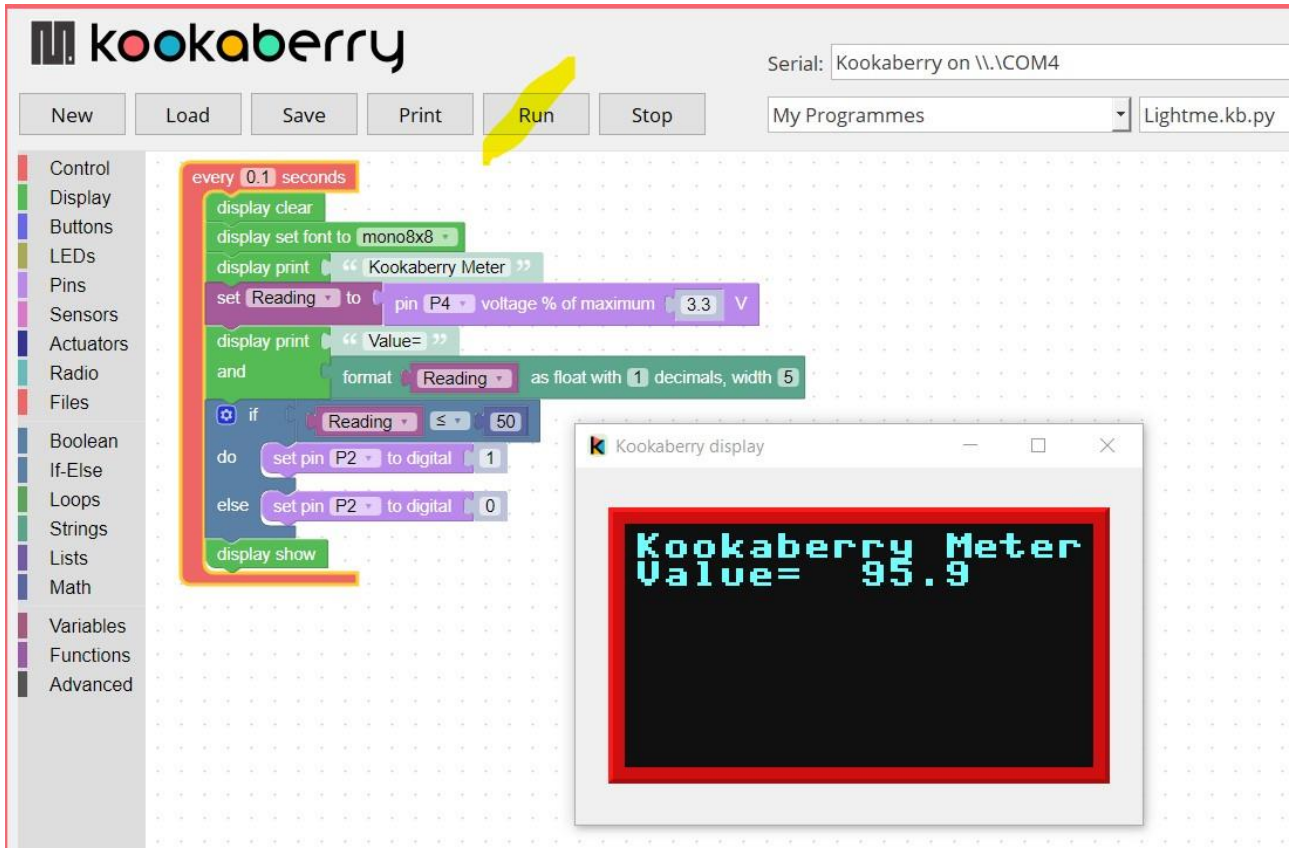
Step 9: Run & Show App

- Connect an ambient light sensor to Pin 4, a LED to Pin 2, and click Run at the top of KookaBlockly window.
- Click on Show display (Top RH corner of window) to display the screen in a separate Teachers' Window (TW).

The TW can be displayed on a class monitor or Smartboard.

The value of the ambient light reading (expressed as a % of maximum) will appear on the Kookaberry's screen.

Make sure that the reading is greater than 50% and then cover the sensor to reduce the reading to less than 50%. The LED should light up.

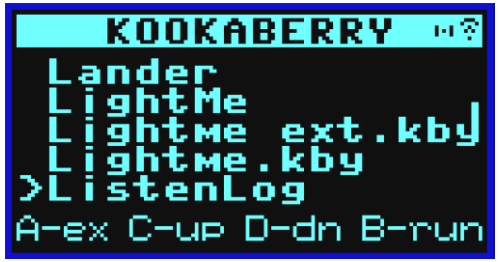


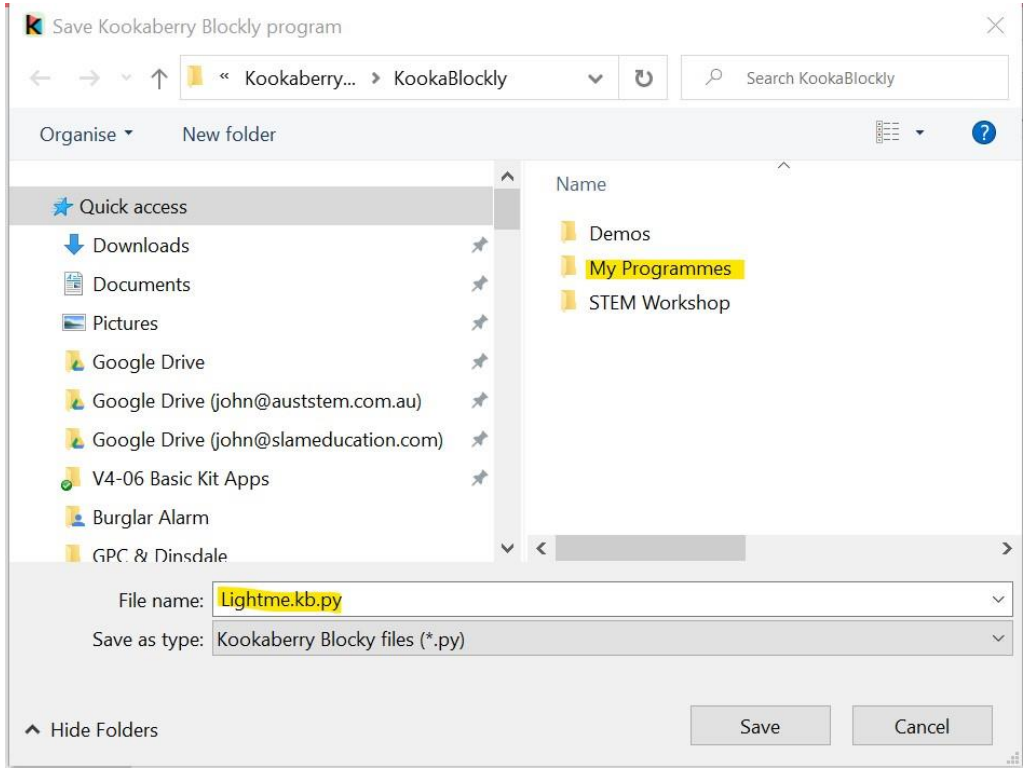
kookaberry

Step 10: Save App and include in Kookaberry menu

- Click on Save at the top of the window; select My Programmes folder and give your programme a title. Make sure that it is in the format *.kby.py and that the **first letter of the name is in capitals.**

This KookaBlockly (kby) file can be copied and pasted into the memory of any Kookaberry where it will show and can be selected from its menu.





NOTE: When retrieving a saved file from the Scripts menu you may sometimes see a blank screen. The programme is there – it is just hiding beyond the top left hand corner of the canvas! Click on the top icon at the bottom right of the canvas to centre it.

Extensions to this Lesson Plan

The following slides introduce three extensions to this lesson plan by adding the following to the Kookaberry's screen

- an "Exit app" instruction
- Text instructing how to exit app.
- An icon showing light is on

These extension exercises can take some time to complete – particularly adding a graphic icon.

Learning Outcomes

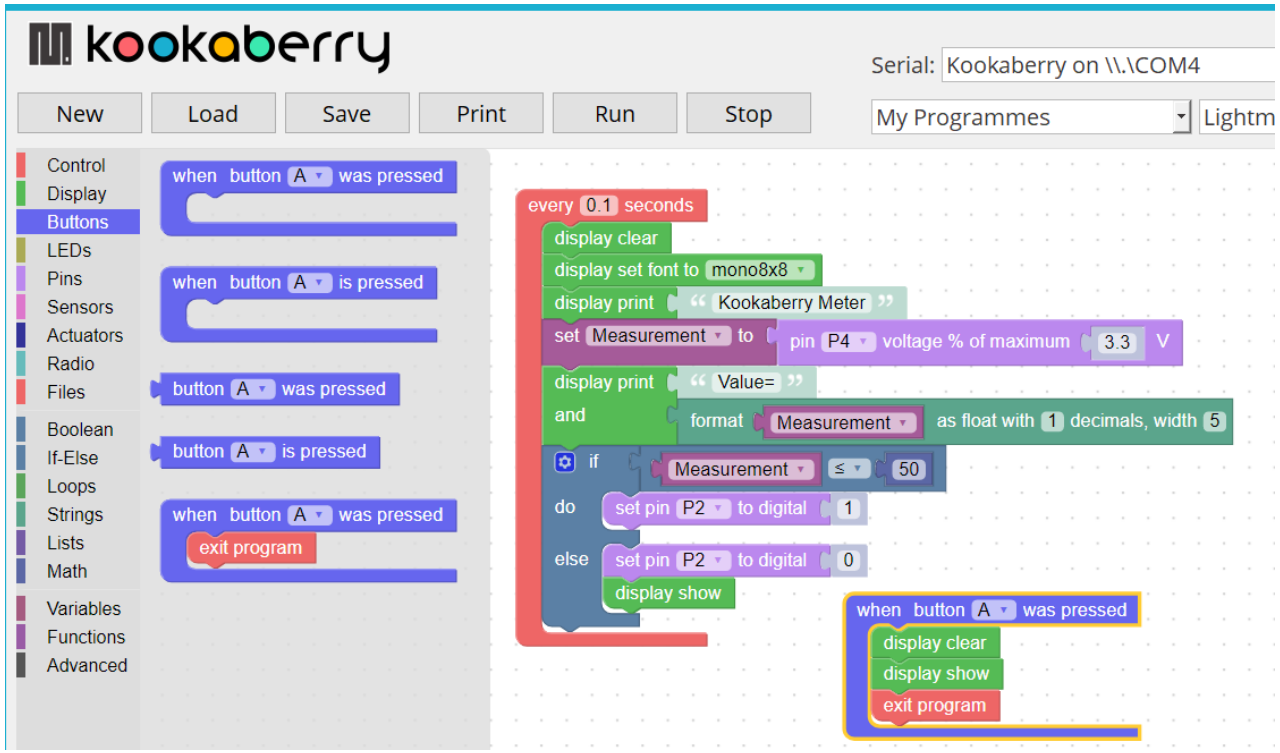
Students will learn how to

- Design and implement "Button" commands
- Position text on the screen by using x and y coordinates.
- Create graphic icons

Extension 1: Add Exit function

- Drag the “when button [A] was pressed” command from the Buttons category onto the canvas.
- Drag the “display clear” and “display show” from the Display category to within the button command block
- Drag the “exit program” block and place it at the bottom of the button block.
- Run the programme and press Button A – the app closes and the screen clears.

Question: Why do we need to include “display clear” and “display show” commands? Try running the app without them included in the button command….



kookaberry

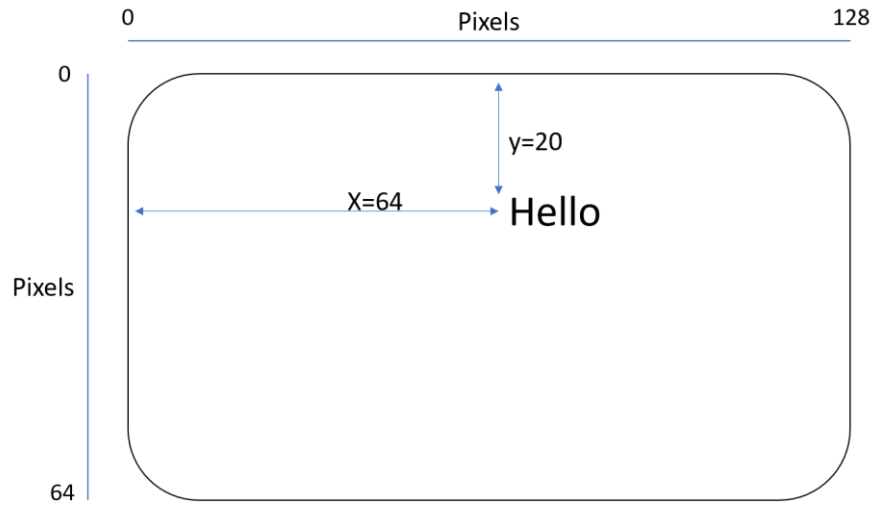
Placing text on the screen

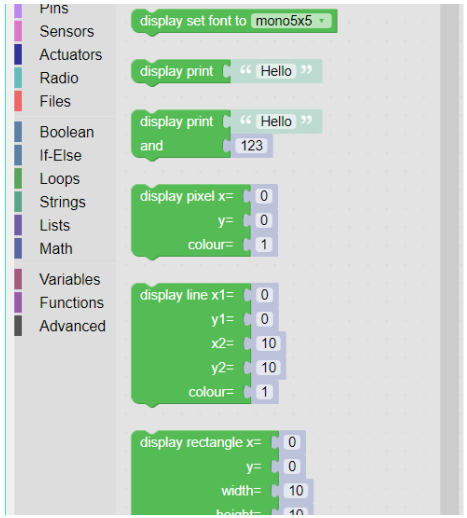
The Kookaberry screen is 128 pixels wide and 64 pixels high

The Display menu in KookaBlockly allows you to place a piece of text or an image on the screen using x and y coordinates.

The x axis is the horizontal dimension, and the y axis is the vertical dimension

Find the appropriate block in the Display menu and move your messages around the screen. Be inventive….



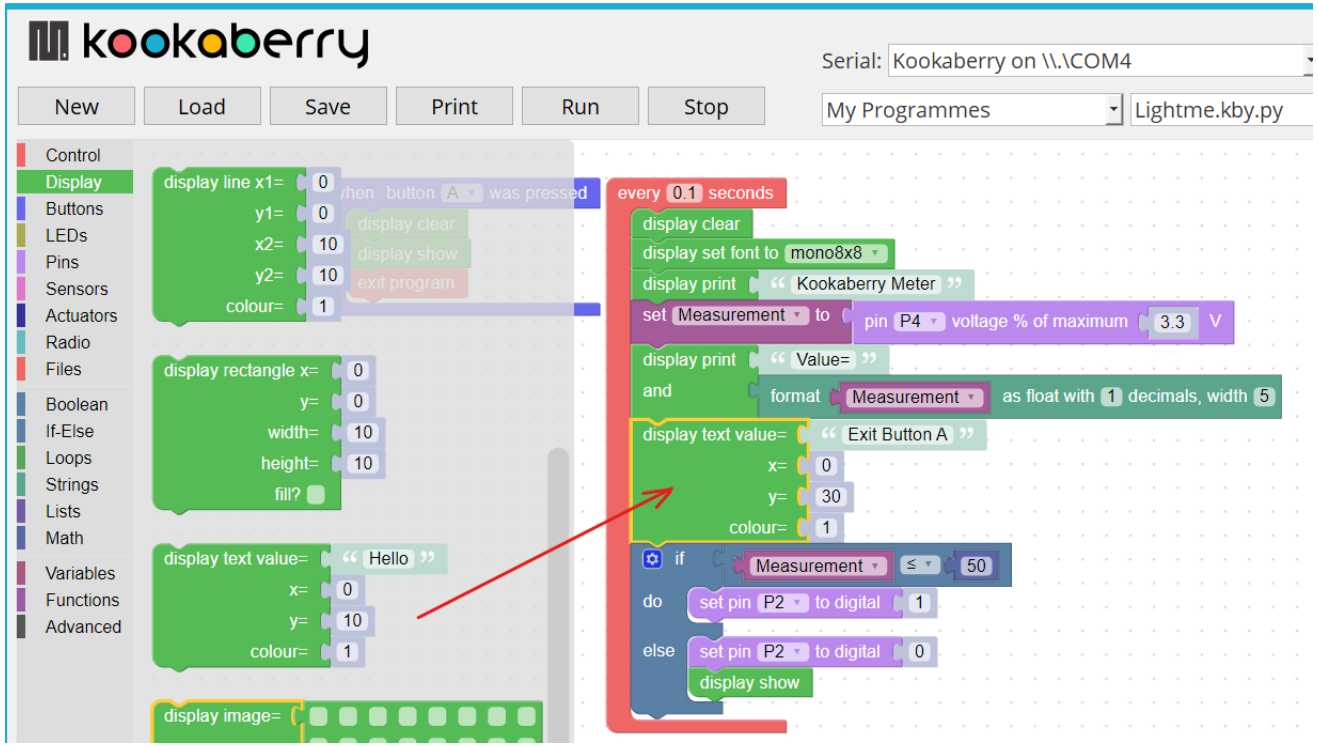


kookaberry

Extension 2: Add Exit text

- Drag the “display text value…..”block from the Display category onto the canvas.
- *It can be located anywhere within the control loop but the position indicated is the most logical.*
- Change the text to “Exit Button A” and the y coordinate to 30
- Run the programme to try it out

Setting the y coordinate to 30, positions the text halfway down the screen



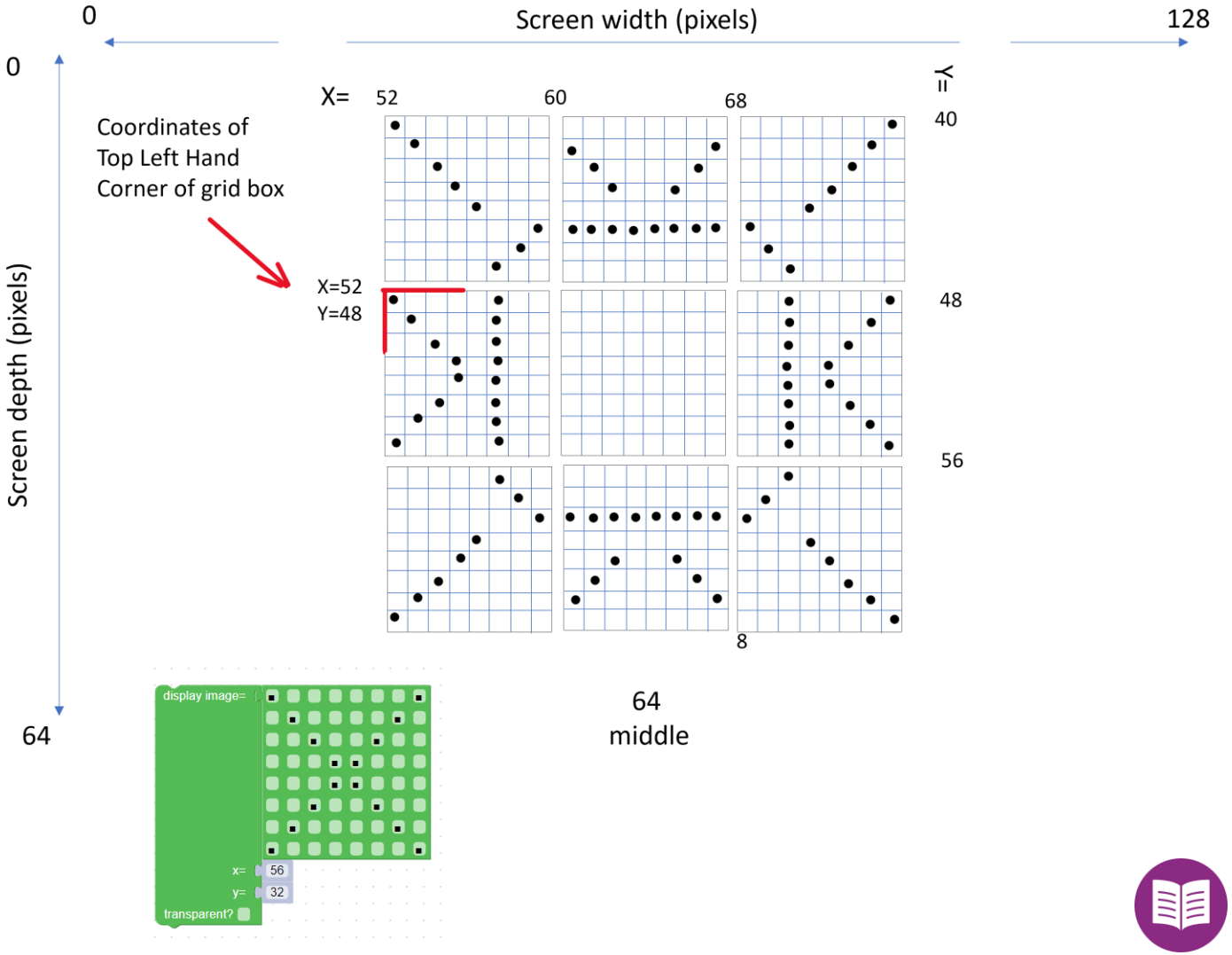
Drawing graphics on the screen

The Display category contains an 8x8 pixel block which is used to construct screen graphics.

Each of the grid elements represents one pixel, and the top left-hand corner of the block can be positioned on the screen using its x and y coordinates

Use your mouse to fill in the little squares to construct your graphic. Join up the blocks using the x and y coordinates.

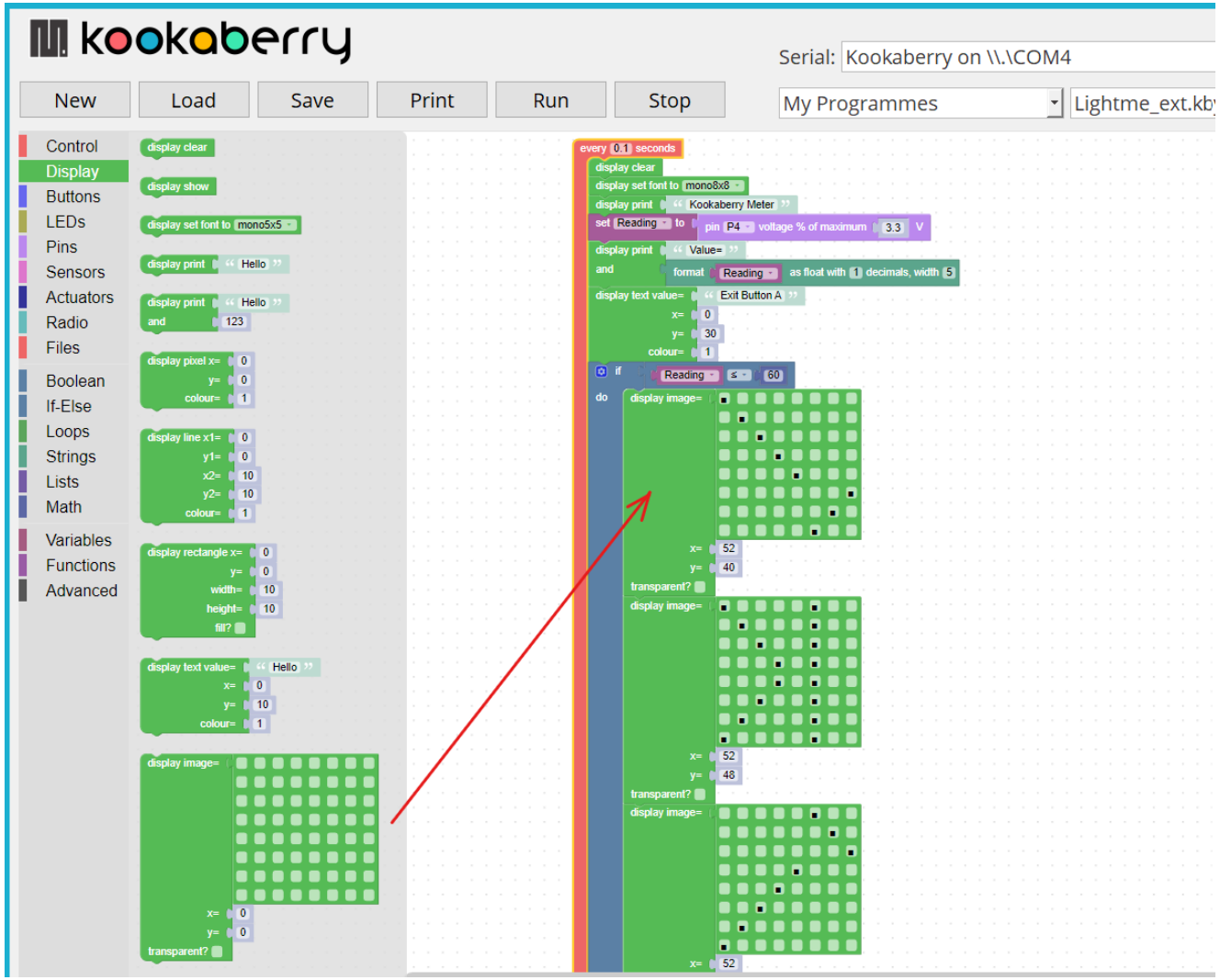
Draw your graphic using squared paper first. The graphic of a light/sun/star is shown to the right.



Extension 3: Add graphic icon

- Drag the “display text value…..”block from the Display category onto the canvas.
- Refer to the drawing you have made on squared paper and place the dots in each of the grid blocks.
- Set the x and y coordinates to correctly position the blocks**. *NOTE**: The y coordinate needs to be more than 40 to place it below the Exit text*
- Place the blocks within the “do” part of the If-Else block. The order does not matter. The icon will appear when the light is below the set threshold.

A graphic as big as this is difficult to get right first time. Run the programme multiple times until you have got all the coordinates right.



kookaberry

Mapping to NSW Curriculum Years 5/6

SCIENCE & TECHNOLOGY

Skills: Design & Production

- Defines problems, and designs, modifies and follows algorithms to develop solutions (ST3-3DP-T)

Technology: Living World

- Explains how digital systems represent data, connect together to form networks and transmit data (ST3-11Di-T) (ACTDIK014)

Mapping to Australian Curriculum Years 5/6

DIGITAL TECHNOLOGIES

Digital Systems

- Examine the main components of common digital systems and how they may connect to form networks to transmit data. (ACTDIK014)

Creating Digital Solutions

Investigating and defining

- Define problems in terms of data and functional requirements drawing on previously solved problems. (ACTDIP017)

Generating and designing

- Design, modify and follow simple algorithms involving sequences of steps, branching, and iteration. (ACTDIP019)

Producing and implementing

- Implement digital solutions as simple visual programs involving branching, iteration, and user input. (ACTDIP020)