Controlling the Lovell Telescope with BBC micro:bits

Coding Workbook

David Whale

dwhale@theiet.org

16th December 2015

1. OVERVIEW

The Lovell telescope is controlled by sending an AZ/EL azimuth (rotation in degrees) and elevation (in degrees) to it's control system over the local area network.

When the telescope is close to a celestial body of interest, an RA/DEC (Right Ascension and Declination) of that celestial body is sent, and the telescope with track it (it will lock onto it by moving the telescope, and then track it as the Earth continues to move).

The control system consists of 3 BBC micro:bits attached to a small model telescope. The telescope is turned and tilted to the correct position and a button pressed to start moving the real telescope to that position. When it is close, another button is pressed to enter tracking mode.

When a signal is received from a pulsar, an animation plays on the 3rd micro:bit, and a speaker beeps.

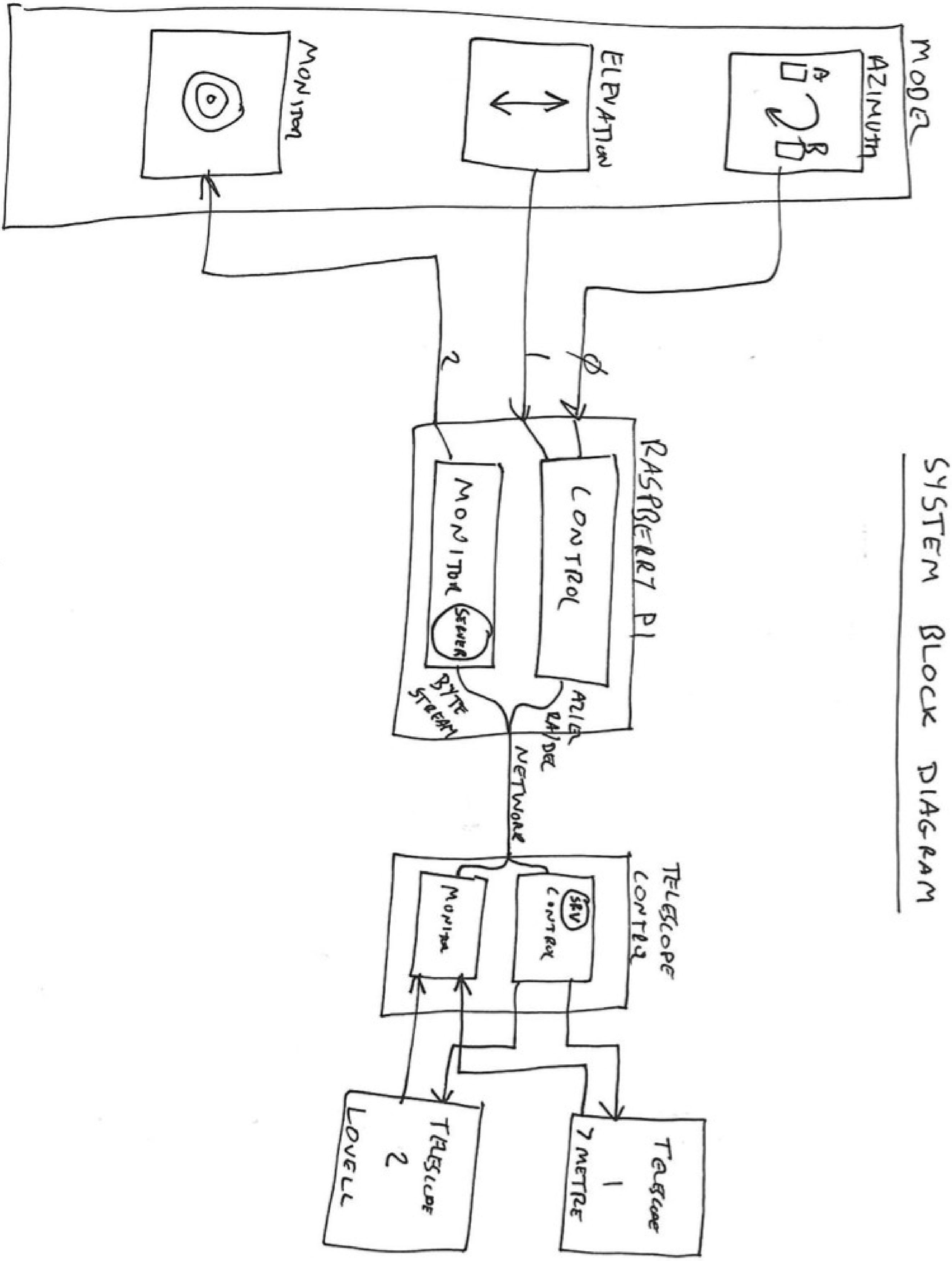
AZ – the compass sets the Azimuth, button A accepts AZ/EL, button B sends RA/DEC

EL – the Y tilt sets the Elevation

MON – the display shows an animation and the speaker beeps when a signal is detected.

2. SYSTEM DIAGRAM

This diagram shows the main components of the system, and how they are connected.



3. micro:bit 1 (AZ)

3.1. Program Design

```
calibrate the compass
loop
    read button A
    read button B
    read compass bearing
    display compass bearing as an image
    build message
    send message
    wait 1 second
end loop
```

3.2. Message Structure

This message will be transmitted via the serial port to the Raspberry Pi on /dev/ttyACM0

A359,1,0

A means it is an Azimuth message.

359 is the compass bearing (0..359) relative to North

The next digit is the state of the A button, 0 = released, 1 = pressed

The next digit is the state of the B button, 0 = released, 1 = pressed

4. micro:bit 2 (EL)

4.1. Program Design

```
loop
    read Y tilt
    show Y tilt on screen
    convert Y from -1024..1024 into 0..90 degrees
    build message
    send message
    wait 1 second
end loop
```

4.2. Message Structure

This message will be transmitted via the serial port to the Raspberry Pi on /dev/ttyACM1

E21

E means it is an elevation message

21 is the elevation from the horizon, in degrees

5. micro:bit 3 (MON)

5.1. Program design

```
loop
  display a dot
  wait for an incoming message
  check if first char is a M
  if it is
    read second char (pulse state)
    if pulse state <> 0
      play half of animation
      beep a short time
      play other half of animation
```

Note: The total time of the animation and the beep must not exceed 200ms

5.2. Message Structure

The following message is sent from the Raspberry Pi to the micro:bit on /dev/ttyACM2

M0

M means it is a monitor message

The next digit is 0 for no signal, 1 for a signal

6. Raspberry Pi Control Program

6.1. Program Design

```
alloc
wait ok/error
stat_timer = 0
loop
    poll ACM0
    if data, check A message, read out bearing → az setpoint, A, B
    poll ACM1
    if data, check E message, read out elevation → el setpoint
    if A
        send AZ/EL setpoint
        wait ok/error
    if B
        send RA/DEC
        wait ok/error
    if stat_timer
        send stat
        stat_timer = 2
    show setpoint, actuals, differences
end loop
on failure, dealloc
```

6.2. incoming message handler

```
if ok, result = ok
if error, result = error
if stat, stat_timer = 0, capture actuals from stat
```

7. Raspberry Pi Monitor Program

7.1. Program Design

```
on incoming:
    filter data
    decrease holdoff
    if not holdoff and got leading edge, P = 1, holdoff for 50 samples
start network server
loop
    if P
        send to monitor (P1)
        P = 0
end loop
```