

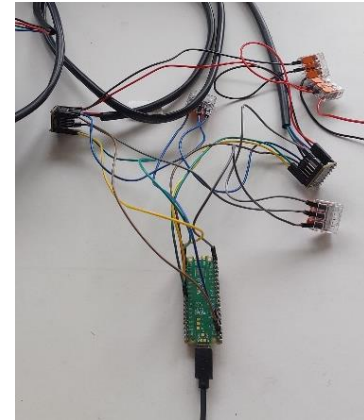
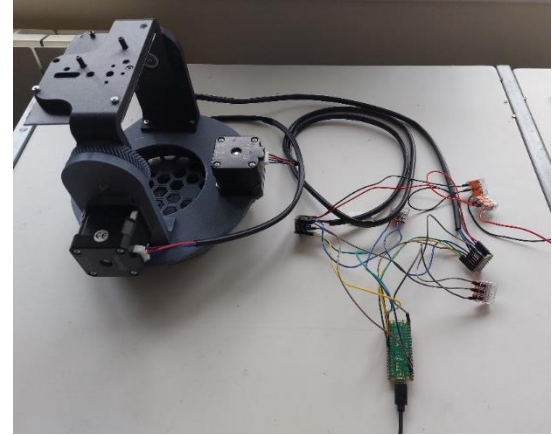


Multi-Spectral Tracking Rig

Josh Asare – 2111739, Jack Heeks – 2111738, Ethan Wilson – 2114187, Robbie Young – 2111907

- Task from TEG to investigate: How a Multi-Spectral Tracking Rig capability can be developed in a cost effective and time efficient manner.

- Designed Electronic control system
 - Raspberry Pi Pico
 - TMC2208 Stepper Motor Drivers
 - NEMA 17 Stepper Motors
- Construction, Assembly & Integration
 - 3D printed PTU parts





Ethan - Firmware

- Developed firmware controlling PTU
- Moves one axis one step at a time
- Each step can be translated into a degree rotation
- Oscillate the step pin the number of steps required
- Split total travel into chunks and alternate between each axis

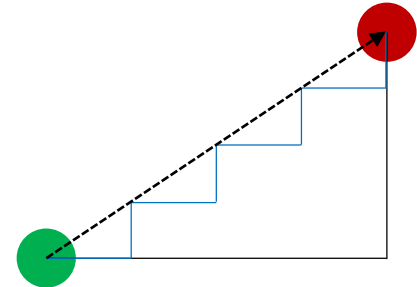
```
#-----Pan Functions-----
#move x-axis by 1 step - Used by all other pan functions
def pan_OneStep():
    pan_tracklocation()
    utime.sleep(delay)
    pan_step.High()
    utime.sleep(delay)
    pan_step.Low()

#Iterates OneStep function a certain amount of times depending on how many degrees to turn
def pan_xSteps(pan_steps):
    pan_report = 0
    global pan_location
    #print location every 100 steps
    for i in range(pan_steps):
        pan_report += 1
        if pan_report == 100:
            pan_report = 0
            print("Pan at "+str(round(pan_location,0))+chr(960))
            pan_OneStep()

#Sets direction of the x-axis rotation and calculates how many steps to achieve a certain degree turn
def pan_moveDegrees(pan_move2):
    if -180 > (pan_location+pan_move2) or (pan_location+pan_move2) > 180:
        pan_move2 = pan_location
        print("Pan limit Reached")

    pan_delta = pan_move2 - pan_location
    if pan_delta > 0:
        pan_dir.High()
    if pan_delta < 0:
        pan_dir.Low()
    pan_delta = -pan_delta

    #degrees to steps - 0.225 degrees per step
    pan_steps = pan_delta / 0.0256
    return pan_steps
```



```

#-----Pan Functions-----
#Move x-axis by 1 step - Used by all other pan functions
def pan_OneStep():
    pan_trackLocation()
    utime.sleep(delay)
    pan_step.high()
    utime.sleep(delay)
    pan_step.low()

#Iterates OneStep function a certain amount of times depending on how many degrees to turn
def pan_xSteps(pan_steps3):
    pan_report = 0
    global pan_location
    #print location every 100 steps
    for i in range(pan_steps3):
        pan_report += 1
        if pan_report == 100:
            pan_report = 0
            print("Pan at "+str(round(pan_location,0))+°")
            pan_OneStep()

#Sets direction of the x-axis rotation and calculates how many steps to achieve a certain degree turn
def pan_moveDegrees(pan_move2):
    if -180 > (pan_location+pan_move2) or (pan_location+pan_move2) > 180:
        pan_move2 = pan_location
        print("Pan Limit Reached")

    pan_delta = pan_move2 - pan_location
    if pan_delta > 0:
        pan_dir.high()
    if pan_delta < 0:
        pan_delta = -pan_delta
        pan_dir.low()
    #degrees to steps - 0.225 degrees per step
    pan_steps = pan_delta / 0.0266
    return pan_steps

```

```

#-----Move Pan & Tilt at same time function-----
#Divides both movements into x chunks. Alternates moving each axis by a chunk until full distance travelled
def both_moveDegrees(pan_move2, tilt_move2):

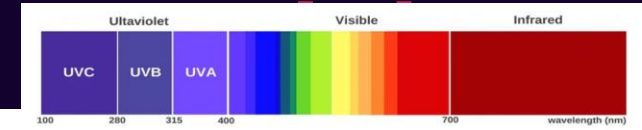
    if (pan_location+pan_move2) and (tilt_location+tilt_move2) >= 45:
        chunks = 200
    else:
        chunks = 30

    pan_steps2 = pan_moveDegrees(pan_move2)
    pan_share = pan_steps2/chunks

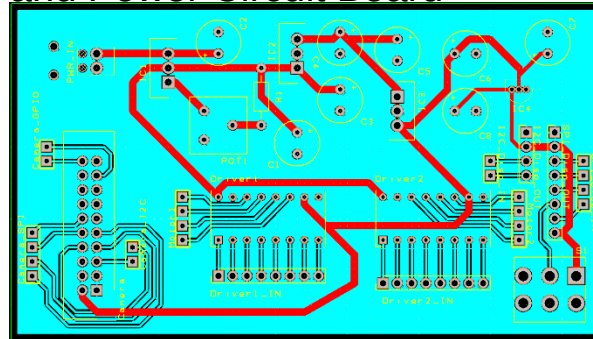
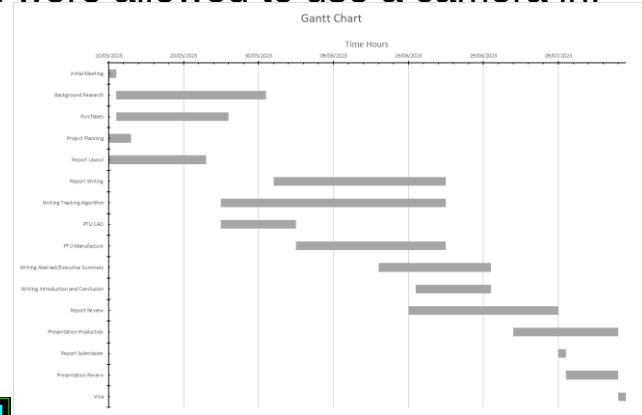
    tilt_steps2 = tilt_moveDegrees(tilt_move2)
    tilt_share = tilt_steps2/chunks

    if debug: print("Debug: pan_share:%s steps:%s tilt_share:%s steps:%s" % (pan_share,pan_steps2,tilt_share,tilt_steps2))
    for i in range(chunks):
        pan_xSteps(pan_share)
        tilt_xSteps(tilt_share)

```



- Workspace Bookings – Finding lab space that we could use and were allowed to use a camera in.
- Booking meetings to check progress
- Gantt Chart and Actual Timeline
- Research
 - UV and IR
 - SBC
 - IR Imager
- Intermediary Communications and Power Circuit Board
- Report Writing
 - Introduction
 - Research
 - Development Methodology
 - Evaluation
 - Conclusion
 - Reflection



- Research
 - Digital image processing
- Development
 - Image processing to remove dimmer background
 - Software to detect bright objects in the camera images
 - UART TX and RX implementation on the Image Processing and Motor Controller boards
 - 3D printed parts of the PTU to enable our timeline to be met
- Report review and Editing





Figure 1: An IR image of a drone in flight

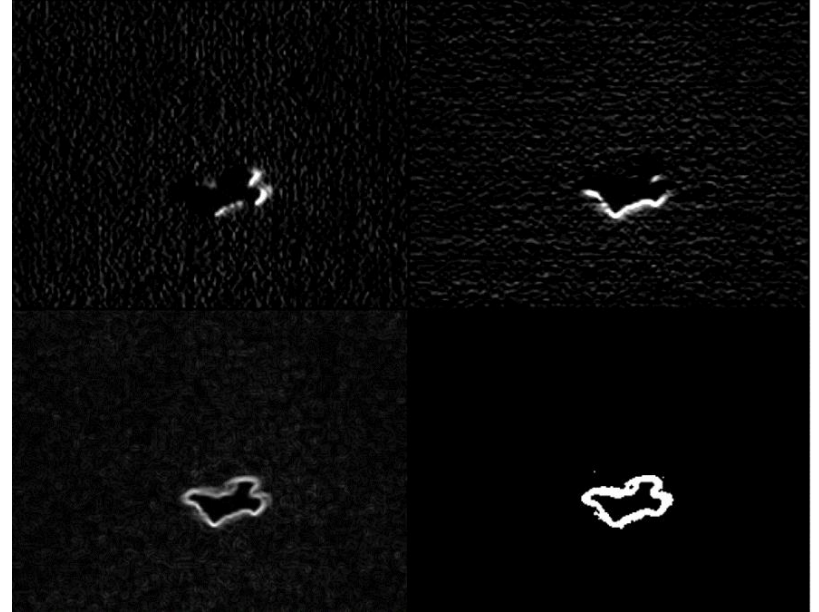
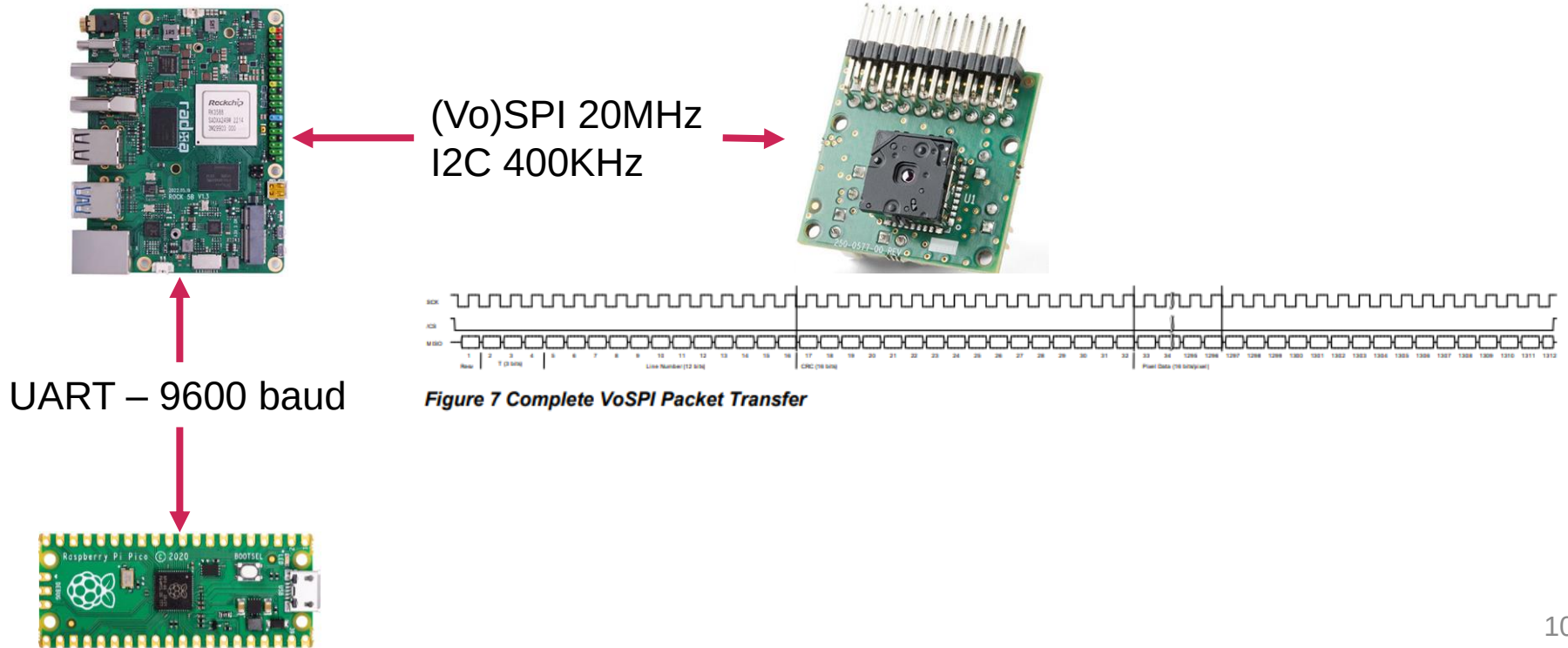


Figure 2: A Sobel operator and threshold applied to Figure 15



- Research
Into 3D printing and the
pan tilt unit
- Development
Mounting a metal plate and
mounting screws for the boards



- Mostly successful
- Total material costs of £664
- Cheaper than COTS PTUs
- Unable to complete conclusive testing due to rushed production

- Project management required
 - Dedicated team leader / project manager
 - Regular project check-ins / meetings
 - Lack of communication throughout the team significantly delayed testing, report writing and reviewing
- Testing was too isolated
- Slip ring implementation skipped due to lack of time, limiting the capabilities of the system

[dstl] The Science Inside

Discover more

