Auto-Pilot

Autonomous control of a remote controlled helicopter

Sarah Currie

Supervisor: James Connan

- Problem Statement
- System Overview
- Results
- Conclusion
- Questions

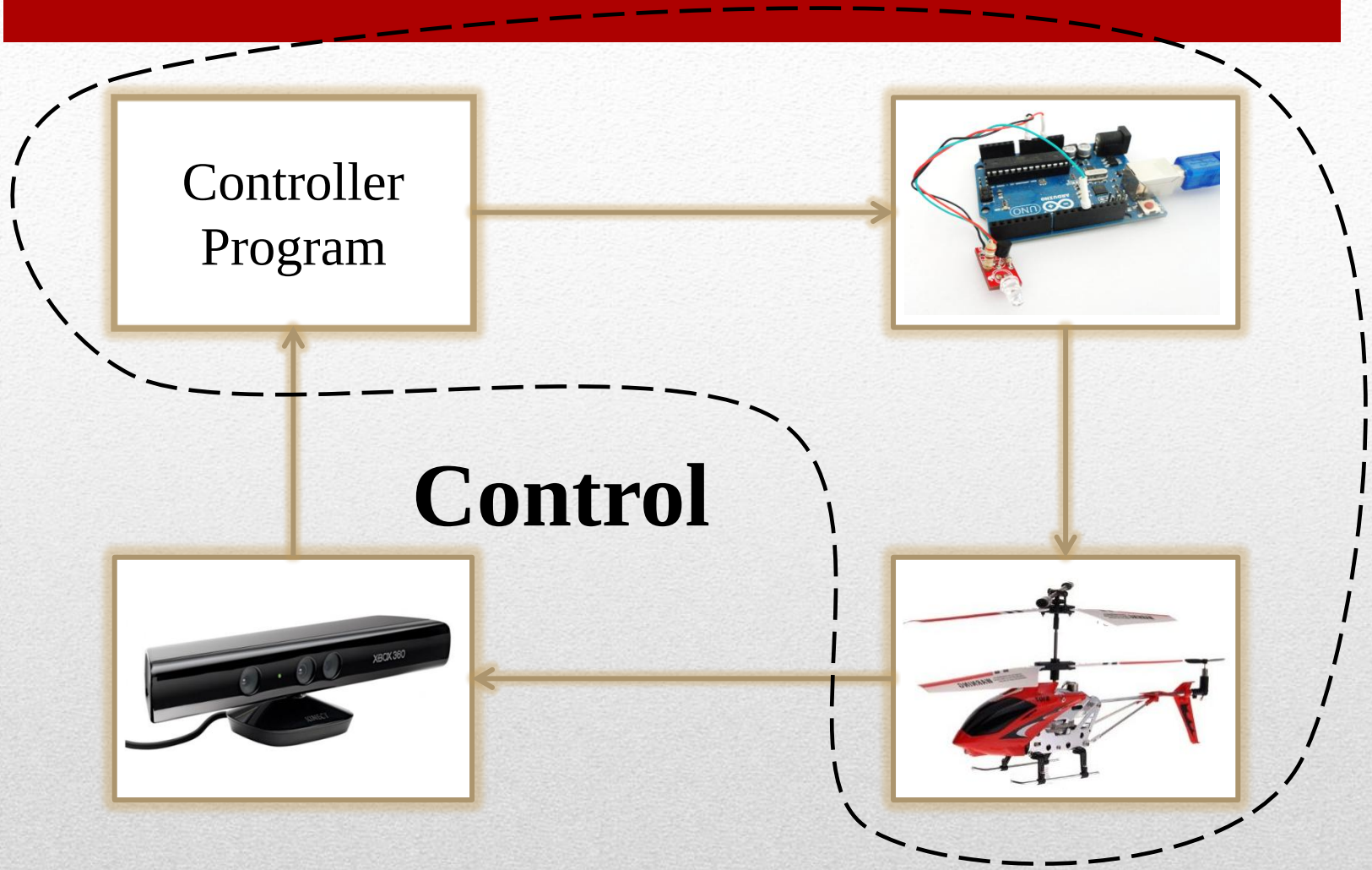
Overview

- Create a system that simulates an auto-pilot for mini R/C helicopter
- Send commands in real-time from PC
- Track helicopter using a camera
- Make helicopter fly in different predefined movements

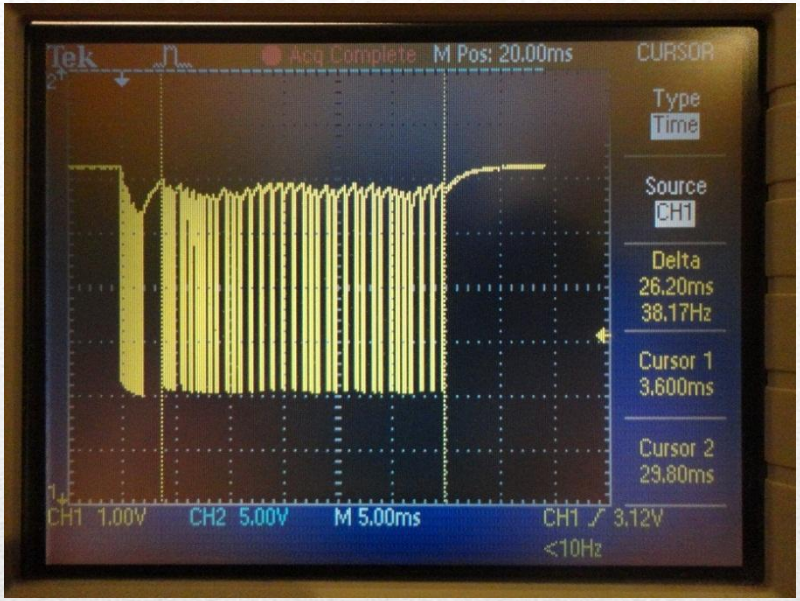
Problem Statement


Desired Position
Controller Program
New Control Values
Infrared Signals
External Disturbances
Tracking Information
Actual Position

Design

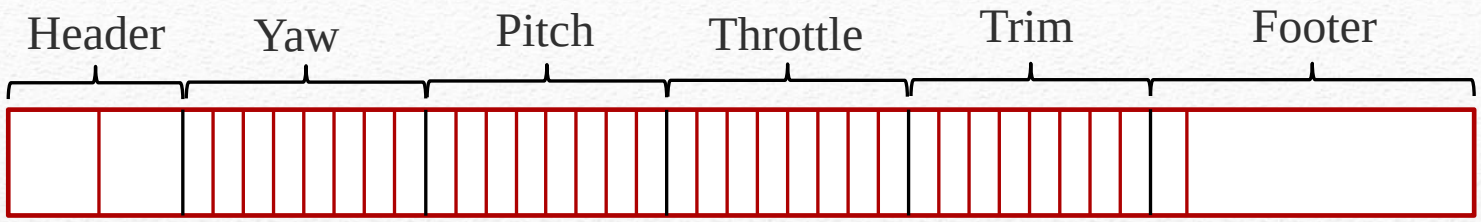


Implementation



- IR signals from original remote measured and analysed
- Reverse engineered IR signals in order to control the helicopter from the computer

Control



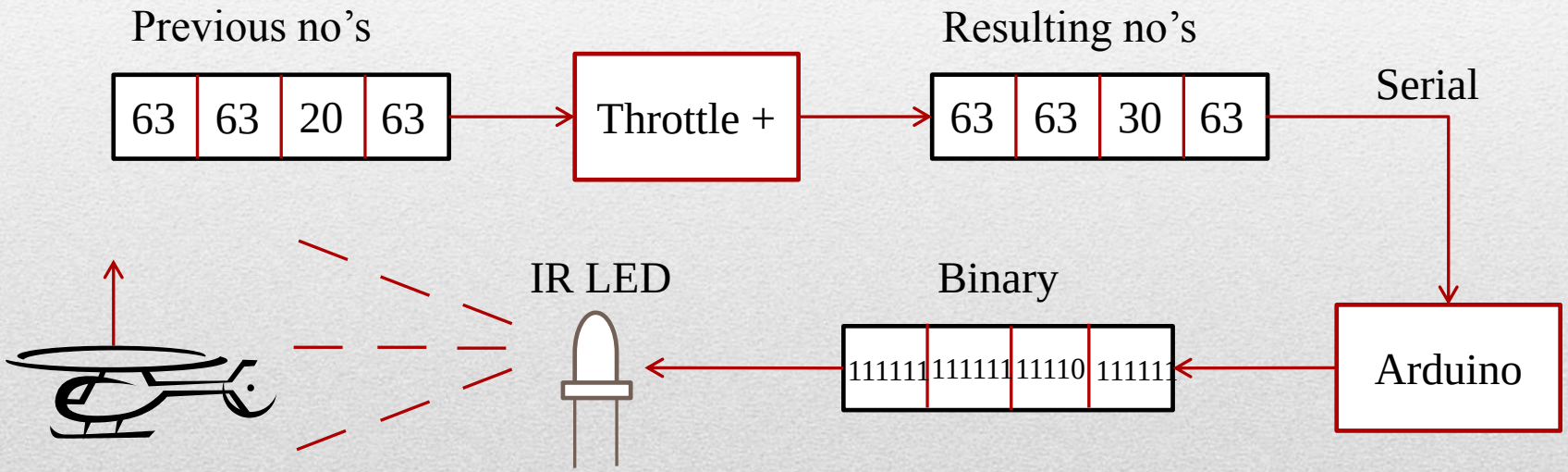
- Each IR packet consisted of a header/start section, a 32 bit data section and a footer/stop section.
- Data Section: 8 bits for each control, represent a decimal number
- Each number ranges between 0 and 127
- The first bit of each number is always set to 0, except the throttle where the first bit indicates the channel

Control

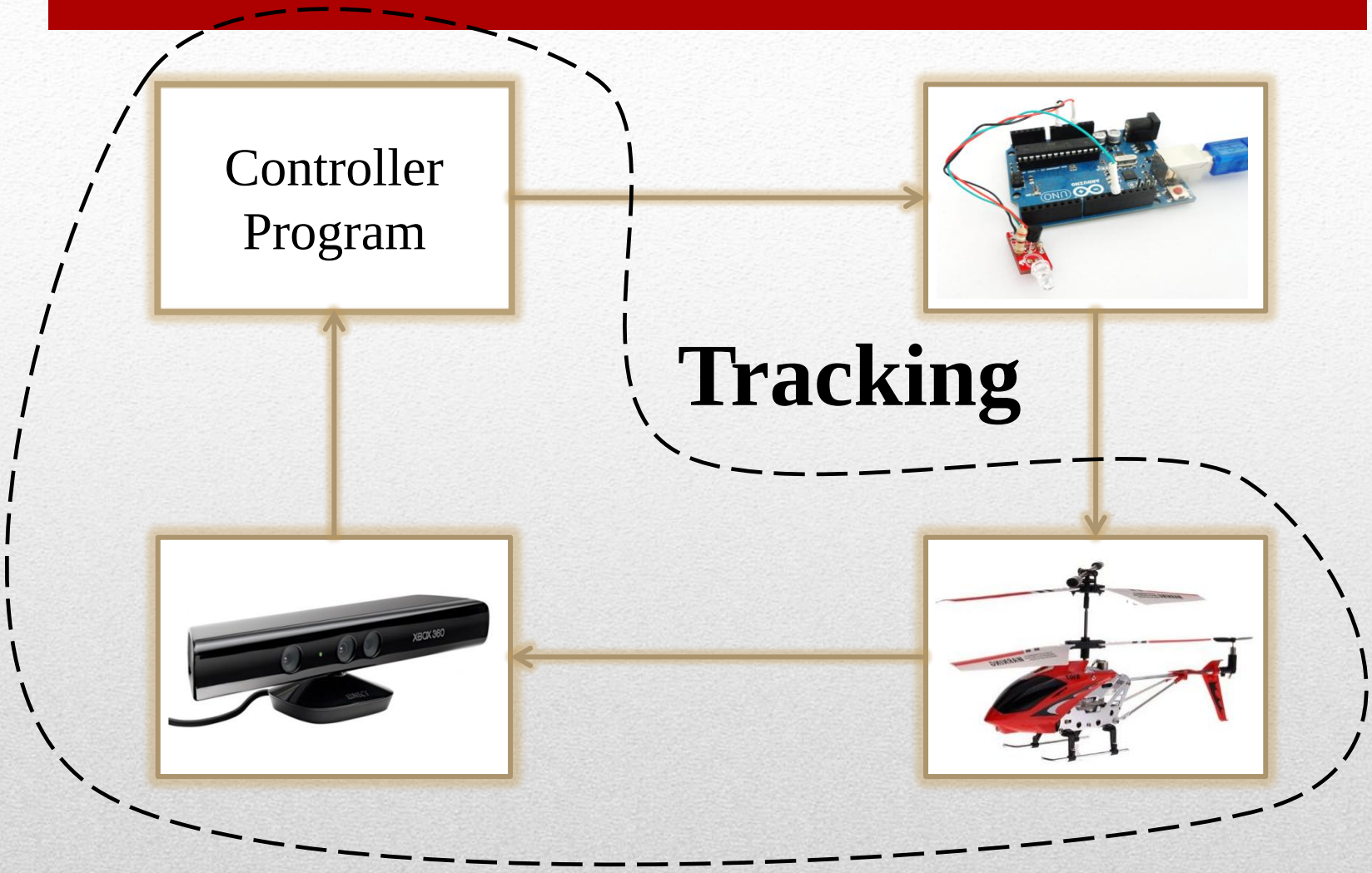
- Program created in Arduino Environment to control IR LED in order to send IR signals
- A C++ program was created to provide a GUI for user with buttons for controlling the helicopter
- Arduino program received serial data from the C++ program containing the new numbers for the controls

Control

- Example: If the user increased the throttle on the GUI, the new throttle number, along with the previous controls numbers would be sent serially to the Arduino program which would convert them to binary and send them in an infrared packet.



Control



Implementation

Tracking

- White LED attached to back of helicopter and front LED made to shine red only
- Brightness of LEDs used for thresholding
- Mask created with threshold results, containing areas of the two LEDs
- The co-ordinates of these areas could then be found
- Orientation of the helicopter was found using the distance between the two LEDs and the depth of the helicopter
- Outcome
  - Helicopter too small for to be assigned a depth value unless East or West, overcome by adding a piece of card to the front of the helicopter to make it appear larger
  - Nothing can be brighter than LEDs in the background, therefore no bright reflections or very white surfaces

Tracking

- CamShift used to make the system more robust
- CamShift tracks objects by their colour histogram
- Creates search window in which the object is predicted to be
- This window was used to reduce search space for LEDs
- Thresholding was only performed within this window
- Outcome
  - Managed to track the helicopter really well in bright scenes. However it still got confused if helicopter flew into the same area as the bright surfaces
  - Extra processing = slightly slower performance

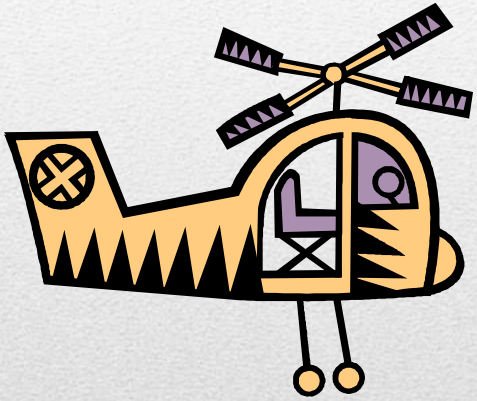
Implementation

Combined Tracking and Control

- The control and tracking programs were combined in order to create a feedback control loop
- A series of tests were performed to test the capabilities and limitations of the system including hovering, flying in a square, navigating to a coloured object
- Limitations of software
  - Helicopter cannot be in the same position as any bright areas in the background
- Challenges of hardware
  - Slight delay of helicopters movements due to overcoming inertia
  - Short battery life causes rapidly degrading performance, making testing difficult and causing varying results
  - Drifting of helicopter made testing difficult

Conclusion

- Even with limitations, the system is still capable of tracking and controlling the helicopter, even if its movements are slightly delayed
- The system fulfils the aim of the project, to control a helicopter autonomously
- Future work
  - Use a larger, more stable helicopter such as a quadropter for higher accuracy and less variation
  - Attach a camera onto a larger helicopter for on-board tracking and navigation
  - Use many, networked helicopters to perform certain movements or tasks



Questions??