Q2: This problem according to me has the following parts which are necessary for maximum points, without going to details.

1) Colour detection of tiles.
2) Navigation → by which I mean which path it must go for a shoot or a pass.
3) Rotation to face the same colour post.
4) Communication protocol between the two bots.
5) Shooting logic according to right of post.
6) Fallback mechanism in case of offset in orientation, path, time or communication.

Assumptions:

1) I assume that passing between 2 robots is 100% irrespective of the direction they are facing. There is no fallback to this, if the shooter robo isn't able to catch the ball considering that time is 60sec and ball path after missing is not computable as it can go outside prescribed arena.

2) Battery life, power is taken care of.

Setup:

My Approch:

- Inetial setup

1) Both robots start at D4 with their +ve x axis forward towards D1 and +y axis leftward.
2) Encoder set to 0, IMU yaw angle 0°.
3) Two constants x, y as co-ordinates of D4
4) Feild map of 4 x 4 field with coordinates of each tile as $\frac{col + row}{2}$. [I have explained later transformation for this]
5) 16 x 16 map containing 0, 1 for valid / non-valid passing between 2 blocks.

— Path planning logic

→ After phase 1 (R_2 is at B3, R1 is at D2.
↳ 16 sec

→ From here there are 4 ways

B_3 colour → Red, Blue, green, yellow.

If Red

R_2 path → B_3 → A_3 → A_2 ⇒ 12 sec

R_1 path → D_2 → C_2 → 4 sec

T = max(12, 4) = 12

16 + 12 = 28 seconds till now

R_2 shoots.

If Blue

R_2 path → B3 → C3 → D_3 → 12sec

R_1 path → D_2 → D1 → 4 secon

T = max(12, 4) = 12

16 + 12 = 28

If Green:

R_2 path → $B_3 \rightarrow C_3 \rightarrow D_3 \rightarrow D_2 \rightarrow$ 12sec

R_1 path → $D_2 \rightarrow C_2 \rightarrow B_2 \rightarrow$ 8

$T = \max(12, 8)$

$T = 16 + 12 = 28$.

If Yellow

R_2 path → $B_3 \rightarrow A_3 \rightarrow$ 4 sec

R_1 path → $D_2 \rightarrow D_3 \rightarrow C_3 \rightarrow$ 8 sec

$T = \max(4, 8)$

$= 16 + 8 = 24$

worst case = 28 seconds. Even if penalty of 5 second is applied + own buffer of 7 seconds = 28 + 5 + 7 = 40 seconds. (This leave 20sec which is enough for fall back)

20 seconds remain for fall back.

→ Fallback mechanism criterion.

I could think 6 instances for fall back mechanism.

→ Robots identifies external blockage detected by proximity sensor < 0.3 or some x after practical implementation

→ Action triggered.

- stopping precomputed path
- A* from current (x, y) to shooting tile / passing tile of the colour detected from BB colour tile.

→2) Teammate blockage (shooter/passer pair)

- Detection
  → Teammate broadcasted position matches with next tiles.

Action
- stop one, block the path, replan.

(little skeptical ↑).

→3) Communication lost between 2 robots.

→ Detection
  → no packet recived for some time 't'.
  assume t > 2.

Action: a) Retry / reset.
b) If B3 tile color is known go to precomputed paths

→ 4) Time

Detection: If no pass && time less than t≤10 in round.

Action: If passmap = 1, the passer passes, so gurantees atleast one or 2 pts. The shooter then navigates to one of 12 shooting pts to shoot if already not in position.

→5) Rotation failure

Detection
Yaw angle error > θ°.
θ° decided after experimentation. Assumption 6°.

Action:
For control Algorithm.
→ Normalize error to ± 180°.
→ using proportional controller
$w = K_p \cdot Error$
K_p is to be tuned.

PID control.

$$W_z = K_p \cdot Error + K_i \cdot \int Error \cdot dt + K_d \cdot \frac{d(Error)}{dt}$$

Then for 4 wheel omni-wheel.

$V_1 = -k\omega_2$ (+, -) signs adjusted for

$V_2 = +k\omega_2$ color of tile.

$V_3 = -k\omega_2$

$V_4 = +k\omega_2$

~~We might consider dropping this step, if~~.

→6) Pre computed path failure.

Detection

→ odometry says current position different from expected position.

Action.

→ A* to nearest valid pose table with remaining target tile avoided.

- Electronic specs

1) Tile colour detection.

Hardware

→ TCS34725 RGB colour sensor

Reasons

→ Economical + IR filter + Stable indoor lighting performance.

→ I^2C Interface.

Placement: This is important because the detection of change of colour from one tile to another is very imp.

- Some distance ~ $\frac{2m}{4}$ under facing ground. Height at around 3cm or as near as possible according to structure.
- Cover to block ambient light for consistent readings.

Calibration: Placing robot over 4 colors to record min-max RGB values. These are stored. This step is necessary for accuracy of colour detection.

Reading & filtering:
only one detection isn't enough, we should read both sensors reading every 10-20ms to get a correct colour reading and the value is matched from min-max values stored beforehand.

2) Navigation + PWM speed control

Hardware specs.
Omni-wheels > mecanums
reason → 1)4x4 tiles with controlled speed, are sufficient & simpler to program.
2) Roller slippage compensation.

Kinematic Model

Variables
V_x → forward / backward speed in robot frame.
V_y → sideway speed in robots frame.
ω → rotation speed about it's axis
x → distance from front + rear wheel
y → " " side wheels.

wheel speeds

$$v_1 = v_x - v_y - \omega(L+W)$$
$$v_2 = v_x + v_y + \omega(L+W)$$
$$v_3 = v_x + v_y - \omega(L+W)$$
$$v_4 = v_x - v_y + \omega(L+W)$$

Some → electronic stackexchange. /modern robotics/northu .ed

But these can be intuitively thought as bunch of vectors cancelling out to give a bunch of vectors whose resultant direction is where we want robot to move.

In short → each wheel's rollers allow sideway slip, so its driving force vectors align with wheel axle orientation.

3) odometry.

AS IMU is used in indoor envo with breaks between rounds my choice is optical encoder, even though it is susceptible to dust as it involves light which might bounce of very small particles also, but it gives accuracy which is more important for short 60 sec rounds.

local frame calc.

$V_x = (V_1 + V_2 + V_3 + V_4)/4$

$V_y = (-V_1 + V_2 + V_3 - V_4)/4$

$\omega = (-V_1 + V_2 - V_3 + V_4)/(4(x+y))$

source → modernrobotics/northwestern.edu

local → global

↳ from first page *

$x\_local = v_x * \Delta t$

$y\_local = v_y * \Delta t$

now as the robot is also rotating, θ should be incorporated in finding, but given specifically they were independent

~~x_local = dx_local~~

~~x_global~~ $\Delta x\_global = x\_local$ ⊗

$X\_global_{new} = x\_local + X\_global$

Similary

⊗ $Y\_global = Y\_local + Y\_global$

~~PWM speed control~~

Now from computed X_global lets Say is 3.34, and Y_global is 2.84 they the cell of x,y gives column gives the row & column numbers.

R ↳ Better way would be take the ceil of x_global and add +1 to reduce drift over the seconds.

Rotation:

The Yaw angles 0 → green, 90° → red, 180 → Yello, 270 → Blue are stored before hand.

Whenever the tile colour change is predicted, the rotation will rotate to corresponding direction. The rotation direction is decided by colourID, the storage of which is gave above.

→ Rotation rule

→ Rotate until a coloured angle is within ±5 for practicality.

error is handled in ~~bac~~ fallback mech.

Communication protocol.

- Using UART because of it range 15m is above the max of field. others have range 1m, 0.2m.

Setup

→ MCU → MCU (micro controller unit) direct wireless serial connection.

→ common ground.

Packet Structure: UART sends Byte streams. So a defined fixed lenght packet with start + a mechanism to detect error.

Packet conentents

[Adress] [sender] [tile ID] [color ID] [Status for pass] [timestamps] ↓ Low + High

→ Sends packets every 100ms.

→ Rollout Backup plan if no message upto t>2.